

**MAGAZINE**

# **BSD**

**FOR NOVICE AND ADVANCED USERS**

## **INFRASTRUCTURE MANAGEMENT**

**ACTIVE DIRECTORY**

WITH SAMBA AND BIND ON FreeBSD

**FreeBSD SERVER MANAGEMENT**

WITH ANSIBLE

**OPENBSD AS A GATEWAY FIREWALL**  
FOR SOHO AND ENTERPRISE NETWORKS

**UNIX BLOG SECTION - MEET UNIX BLOGGERS**

**OPNsense**

**TAKING A LOOK AT SMARTOS CONTAINERS**

VOL. 11 NO. 02

ISSUE 02/2017 (90)

ISSN 1898-9144

Dear Readers,

The BSD team and I are delighted to declare a new coming issue of BSD Magazine. We completed this issue to include a large pile of tutorials and practice rich articles for you to construct up your open-source skills and cognition. This time, we publish more on Infrastructure Management, and I hope you'll like this batch of articles. I would like to thank the authors of this BSD issue as all of them did great work and wrote insightful, high-level and technical articles in a very short time. Thank you. Also, I would like to mention here that those who did not write for this issue are currently working on the next issues of the BSD magazine and writing the articles that will be devoted to security and cloud computing. Thank you all for your time and willingness to help. I've started with this BSD issue after a long time break and I am happy that I can meet so many open-minded, knowledgeable, hard-working and trustworthy people. I received and I am still receiving many emails from you, and you make my day everytime I go through them. Write anytime. Feel free to ask any questions regarding the BSD Magazine or send your opinion on what we can do to make it better. Working with you was a great pleasure.

Our common ultimate goal was and still is to supply our readers with the knowledge and skills they need in their professional careers. We are happy to take your suggestions of future articles and tutorials what you need most to see in our magazine, and thus along.

Let's take a look at what you will encounter in this issue of BSD. Our experts will instruct you on how to deal with Virtual Firewall and how to install OPNsense on Bhyve. Thereafter, we will be heading to OpenBSD as a Gateway Firewall for SOHO and Enterprise Networks. In this issue, you will read more on SmartOS Containers. Last but not least, we'll share more concerning a FreeBSD Server Management with Ansible, Active Directory with Samba and BIND on FreeBSD, OPNsense, and much more. Make sure to check out this issue for more articles and tutorials.

If you want to start a real life open-source journey with our rich-content publications, or to get in contact with our team, feel free to write to us.

Best regards,

Ewa & The BSD Team

*PS. As always, we invite other experts, companies, reviewers for collaboration for future work and issues.*

# MAGAZINE BSD

**Editor in Chief:**

Ewa Dudzic

[ewa@bsdmag.org](mailto:ewa@bsdmag.org)

**Contributing:**

Bob Cromwell, David Rodriguez, Carlos Antonio Neira Bustos, Antonio Francesco Gentile, Abdorrahman Homaei, Randy Ramirez, Benjamin Wright, Tom Ryder, Vishal Lambe, Mikhail Zakharov, Pedro Giffuni, David Carlier, Albert Hui, Marcus Shmitt, Aryeh Friedman and Rob Somerville.

**Top Betatesters & Proofreaders:**

Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Katherine Dizon and Mark VonFange.

**Special Thanks:**

**Denise Ebery**

**Annie Zhang**

**Senior Consultant/Publisher:**

Paweł Marciniak

**Publisher:**

Hakin9 Media SK,  
02-676 Warsaw, Poland Postepu 17D Poland  
worldwide publishing

[editors@bsdmag.org](mailto:editors@bsdmag.org)

[www.bsdmag.org](http://www.bsdmag.org)

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: [editors@bsdmag.org](mailto:editors@bsdmag.org)

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.



## [News](#) 3

*BSD Team*

This column presents the latest news coverage of events, product releases, and trending topics.

## [FreeBSD](#)

### [Active Directory with Samba and BIND on FreeBSD](#) 4

*Bob Cromwell*

Bob will advise you on how to run an Active Directory service on FreeBSD. The goal is to explain the new or surprising parts to people from the UNIX world. The Samba domain provisioning itself is surprisingly easy and is documented nicely on the Samba website.

### [FreeBSD Server Management with Ansible](#) 8

*David Rodriguez*

David will mainly focus on Ansible's key features and setup, how Ansible fits into the modern DevOps movement and some of the decisions you will make in designing a FreeBSD server infrastructure. Using FreeBSD as a start, the large number of ports and packages available along with a tool like Ansible enables you to build a large-scale infrastructure quickly.

### [Taking a Look at SmartOS Containers](#) 13

*Carlos Antonio Neira Bustos*

There are a lot of remarkable things to do with SmartOS as a home server. For instance, run containers for development or production in your company, for speed up development using containers, game servers etc. Carlos will start with SmartOS containers in his article.

## [OpenBSD](#)

### [OpenBSD as a Gateway Firewall for SOHO and Enterprise Networks](#) 21

*Antonio Francesco Gentile*

In Antonio's article, you'll see that OpenBSD makes possible the creation of a Gateway Router Firewall and a multi VPN concentrator. The level of security it can give is very high, both for SOHO and enterprise infrastructures; therefore, OpenBSD proves to be a great alternative to using a dedicated and expensive hardware equipment, plus it's open-source.

## [OPNsense](#)

### [OPNsense](#) 38

*Abdorraahman Homaei*

Abdorraahman in his article explains what OPNsense is and presents more about OPNsense vs. PFsense. He describes how to deal with Virtual Firewall, how to install OPNsense on Bhyve, and what OPNsense Mandatory Configuration is.

## [Meet Unix Bloggers](#)

### [Kill a long running process in Unix](#) 42

*Vishal Lambe*

### [Shell From vi](#) 46

*Tom Ryder*

## [Interview](#)

### [Interview with Benjamin Wright](#) 50

*Ewa & The BSD Team*

## [Column](#)

### [Infrastructure Management](#) 52

*Randy Ramirez*

# BSD Certification

**The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.**

## WHAT CERTIFICATIONS ARE AVAILABLE?

**BSDA: Entry-level certification** suited for candidates with a general Unix background and at least six months of experience with BSD systems.

**BDSP: Advanced certification** for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

## WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:  
<https://register.bsdcertification.org/register/payment>

## WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:  
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:  
<https://register.bsdcertification.org/register/get-a-bsdcg-id>



## Qt 5.7.1, KDE Frameworks 5.31 Available

The KDE-FreeBSD team announced that [Qt 5.7.1](#) and [KDE Frameworks 5.31](#) in the official FreeBSD ports tree is available. This release has one new KDE Framework, Kirigami 2. It also messages the Qt4 and Qt5 ports. And, it adds misc/qtchooser which allows developers and sysadmins to manage multiple concurrent Qt installations. They advise users to consult the UPDATING entry for 20170218.

<https://freebsd.kde.org/>

## Rust-based Redox OS 0.0.6 Released

Redox OS, a microkernel OS written in Rust, has just released version 0.0.6 which includes bug fixes and an update to Rust.

<https://github.com/redox-os/redox/releases/tag/0.0.6>

## Unix History Repository on Github

The history and evolution of the Unix operating system is made available as a revision management repository, covering the period from its inception in 1970 as a 2.5 thousand line kernel and 26 commands to 2017 as a widely-used 27 million line system. The 1.1GB repository contains about half a million commits and more than two thousand merges.

The repository employs a Git system for its storage and is hosted on GitHub. It has been created by synthesizing with custom software 24 snapshots of systems developed at Bell Labs, the University of California at Berkeley, and the 386BSD team, two legacy repositories, and the modern repository of the open-source FreeBSD system. In total, about one thousand individual contributors are identified, the earlier ones through a primary research.

The data set can be used for empirical research in software engineering, information systems and software archaeology.

<https://github.com/dspinellis/unix-history-repo>

## BSDCan 2017

A BSD conference will be held in Ottawa, Canada. It will be a technical conference for people working on and with 4.4BSD based operating systems and related projects.

It includes: tutorials: 7-8 June 2017 (Wed/Thu) and conference: 9-10 June 2017 (Fri/Sat) trials.

<https://www.bsdcn.org/2017/>

## The Envious Pedigree of UNIX® AND POSIX®

But today's breakthroughs would not have been possible without what came before them—a fact we sometimes forget. Mainframes led to personal computers, which gave way to laptops, then tablets and smartphones, and now a myriad of gadgets.

<https://blog.opengroup.org/2016/08/17/the-enviable-pedigree-of-unix-and-posix/>

## AsiaBSDCon 2017

March 9 - 12, Tokyo University of Science, Tokyo, Japan.

AsiaBSDCon is a conference for users and developers on BSD based systems. The conference is for anyone developing, deploying and using systems based on FreeBSD, NetBSD, OpenBSD, DragonFly BSD, Darwin and MacOS X. AsiaBSDCon is a technical conference with an aim of collecting the best technical papers and presentations available to ensure that the latest developments in our open-source community are shared with the widest possible audience.

<https://2017.asiabsdcon.org/>

## EuroBSDCon 2017

September 21 - 24, 2017, Paris, France

EuroBSDCon is the premier European conference on the open-source BSD operating systems, attracting about two-hundred and fifty highly skilled engineering professionals, software developers, computer science students, professors, and users from all over Europe and other parts of the world. The goal of EuroBSDCon is to exchange knowledge about the BSD operating systems, facilitate coordination and cooperation among users and developers.

<https://2017.eurobsdcon.org/>

## Community BSD Events Website

The BSD Events website was created for a complete up to date list of all events for the entire BSD community, including conferences and other get-togethers. Dates and locations for upcoming BSDCG exam events are also included.

[www.bsdevents.org](http://www.bsdevents.org)



# Active Directory with Samba and BIND on FreeBSD

**I was working on some projects that needed an Active Directory server. These were proof-of-concept projects, figuring out how to integrate Linux and BSD servers with Windows desktops and, potentially, some Windows servers. That means using Active Directory.**

Here is how to run Active Directory service on FreeBSD. The goal is to explain the new or surprising parts to people from the UNIX world. The Samba domain provisioning itself is surprisingly easy and is documented nicely on the [Samba](#) website. I have much more [detail on my site](#) if you want to go deeper into the whole project.

## What is Active Directory?

Active Directory or AD is Microsoft's bundle of DNS, LDAP and Kerberos. BIND is the Internet-standard DNS server, and Samba 4 includes LDAP and Kerberos, so we have the needed pieces. Other Microsoft-specific services are common in AD environments like MS-SQL, Exchange, AD Certificate Service, etc., but Microsoft is adamant about the AD server running nothing but AD.

Samba uses the Heimdal implementation of Kerberos. It will be compatible with the original MIT code, or with Microsoft's version. Windows systems, both clients and Kerberized services, won't realize that they aren't communicating with a native Microsoft Active Directory server.

## Getting Started — FreeBSD on Raspberry Pi

The goal was experimenting and developing, not deployment with enterprise-scale performance. So, I used [RaspBSD](#), a FreeBSD image for the Raspberry Pi and other very low *cost/size/power* platforms.

The first Samba-specific requirement is adding lines to `/etc/hosts` to map each IPv4 and IPv6 address to the hostname and FQDN.

I also added syslog, configuring it to send nothing to files and everything across the net to a log collector.

## Stumbling Block #1

I encountered my first problem when I added BIND and Samba. The RaspBSD repository has the following packages; [bind99](#), [bind910](#), and [bind911](#), providing BIND version 9.9, 9.10, and 9.11, respectively, and [samba42](#) and [samba43](#) providing Samba 4.2 and 4.3. But you can't install the very latest of both!

Samba uses *dynamically loadable zones* (or BIND9\_DLZ) accessible through the AD schema. Samba needs the appropriate shared library for the installed version of BIND. Samba 4.3 only supports BIND 9.10, BIND 9.11 requires at least Samba 4.5.2. So, it was `bind910` and `samba43`.

## Stumbling Block #2

The second problem was very BSD-specific. The RaspBSD image of FreeBSD has NFSv4 ACLs enabled within the superblock of the root file system. If a flag is set in the superblock, it is used at initial mount time regardless of `/etc/fstab` contents.

```
# mount | grep s2a
```

```
/dev/mmcblk0s2a on / (ufs, local, noatime, journaled soft-updates, nfsv4acls)
```



Samba used to support NFSv4 ACLs, as long as you specified that during the initial deployment. However, Samba presently requires POSIX ACLs to protect /var/db/samba4/sysvol/. The superblock must be modified. I did it interactively after manually stopping almost all processes:

```
# sync
# sync
# mount -f -u -o ro /
# dumpfs / | grep flags
```

```
flags soft-updates+journal nfsv4acls
```

```
# tuneefs -N disable /
```

tuneefs: NFSv4 acls cleared

tuneefs: filesystem reloaded

```
# tuneefs -a enable /
```

tuneefs: POSIX 1.e ACLs set

tuneefs: filesystem reloaded

```
# dumpfs / | head -22
```

magic 19540119 (UFS2) time Fri Feb 17 15:42:56 2017

superblock location 65536 id [ 5887d304 7cce2791 ]

ncg 130 size 7787248 blocks 7540727

bsize 32768 shift 15 mask 0xffff8000

fsize 4096 shift 12 mask 0xfffff000

frag 8 shift 3 fsbtodb 3

minfree 8% optim time symlinklen 120

maxbsize 32768 maxbpg 4096 maxcontig 4 contigsumsize 4

nbfree 883015 ndir 5571 nifree 3844183 nffree 471

bpg 7493 fpg 59944 ipg 30080 unrefs 0

nindir 4096 inopb 128 maxfilesize 2252349704110079

sbsize 4096 cgsize 16384 csaddr 1920 cssize 4096

sblkno 24 cblkno 32 iblkno 40 dblkno 1920

cgrotor 0 fmod 0 ronly 0 clean 0

metaspace 2392 avgfpdir 64 avgfilesize 16384

```
flags soft-updates+journal acls
```

fsmnt /

volname swuid 0 providersize 7787248

cs[.cs\_(nbfree,ndir,nifree,nffree):

(1635,451,25991,4) (15,441,24276,7) (57,440,25955,109) (190,443,23526,23)

(1949,30,23230,41) (2266,150,24514,113) (4308,356,26422,58) (3137,218,27997,61)

```
# reboot
```

Back on Track

After that, things went smoothly. I added records on the BIND master server to define the new FreeBSD system, including the peculiar-looking SRV or Service records. The zone file contained the following:

[... lines deleted ...]

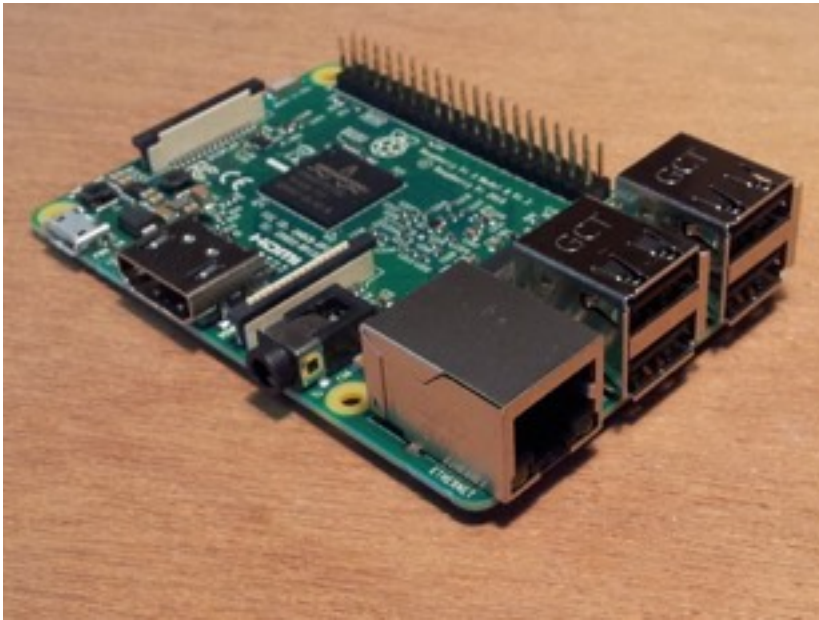
freebsd IN A 10.1.1.235

freebsd IN AAAA fc00::ba27:ebff:fe41:b9ae dc IN CNAME  
freebsd ;;; \_service.\_protocol.DNSdomain IN SRV priority weight  
port target \_ldap.\_tcp.corp.example.com. IN  
SRV 0 0 389 freebsd \_ldap.\_udp.corp.example.com. IN  
SRV 0 0 389 freebsd \_kerberos.\_tcp.corp.example.com. IN  
SRV 0 0 88 freebsd \_kerberos.\_udp.corp.example.com. IN  
SRV 0 0 88 freebsd \_kpasswd.\_tcp.corp.example.com.  
IN SRV 0 0 464 freebsd \_kpasswd.\_udp.corp.example.com.  
IN SRV 0 0 464 freebsd

[... lines deleted ...]







The AD-specific names are `"_service._protocol._dns-domain"`, and the records contain priority and weight for load balancing. See [Microsoft TechNet](#) for more information concerning the service records. Then, I set up the FreeBSD system as a BIND slave server and tested its DNS functionality.

## Setting up Samba

Start by checking where things will go.

```
# smb -b | grep /
```

Configuration went in `/usr/local/etc/`, programs in `/usr/local/bin`, logs in `/var/log/samba4/` (I'll change that) and all the Samba (and LDAP and Kerberos) details in `/var/db/samba4/`. The ease of doing this was a nice surprise! You can do it interactively, or all on one line. Either way, make sure you include [RFC 2307](#) support so that AD can record UNIX attributes like UID, home directory, etc. Here is an example of a non-interactive syntax.

```
# samba-tool domain provision --use-rfc2307 \
    --realm=CORP.EXAMPLE.COM \
    --domain=CORP \
    --server-role=dc \
    --dns-backend=BIND9_DLZ
```

Read through the narrative output and record the automatically generated administrator password and domain SID.

## Remaining Manual Steps

Check `/var/db/samba4/private/named.conf` to verify that it found the appropriate BIND9\_DLZ library. Thereafter,

add a line to your main BIND configuration file to include this Samba `named.conf` file. Also, create a symbolic link `/etc/krb5.conf` pointing to the newly generated file `/var/db/samba4/private/krb5.conf`. Another AD-specific configuration file is `/usr/local/etc/smb4.conf`. After some testing [illustrated on my site](#), I added some lines to `/etc/rc.conf`:

```
named_enable="YES"
samba_server_enable="YES"
ntpd_enable="YES"
ntpd_sync_on_start="YES"
syslogd_enable="YES"
```



Reboot, and it works! Now, we can define AD groups and users with `samba-tool`, and Kerberos principals, authorizations and cryptography requirements with `kadmin`.

## Now AD is running!

I have what I need for my projects, and I hope you found this useful! There's a lot more information about Kerberos administration at [MIT](#) and [Heimdal](#).

### About the Author



Bob Cromwell has been using OpenBSD since, well, not sure how long... Some time in the late 1990s. He's used Linux since you downloaded 40+ floppy images, some time around 1993-1994.

Before that he had used UNIX, SunOS and forms of BSD, at Purdue since the mid 1980s. He got a BSEE at Purdue back then, worked at the university, grad school, Ph.D. in electrical and computer engineering, has done consulting since 1992. He's taught courses for Learning Tree International since the mid 1990s, and has written courses for them since the late 1990s.



# EMERGENCY CURING

for Windows workstations and servers  
including those running other anti-virus software



## FUNCTIONS:

- Cures Windows workstations and servers.
- Verifies the quality of the anti-virus software currently in use.

## FEATURES:

- Dr.Web CureIt! doesn't require installation and doesn't conflict with any known anti-virus; consequently there is no need to disable the anti-virus currently in use to check a system with Dr.Web CureIt!.
- Improved self-protection and an enhanced mode for more efficient countermeasures against Windows blockers.
- Dr.Web CureIt! is updated at least once an hour.
- The utility can be launched from removable media including USB storage devices.

## LICENSING FEATURES:

The utility is available for free when used for non-business purposes.



# FreeBSD Server Management with Ansible

**Configuration Management software is currently a thriving market and full of excellent options. Tools like Chef, Puppet, SaltStack, CFEngine and Ansible can help to configure a large number of servers. Each tool has subtle differences that translate to pros and cons. Some of the tools run on Ruby while some run on either Python or even C.**

Some use a ‘push’ model where the configuration changes are forcefully pushed to the infrastructure. Other tools use a ‘pull’ model where the individual nodes in the infrastructure reach out on a timed interval and pull down changes.

## Why Ansible?

As a personal choice, I prefer to use Python over Ruby, and I like to see the instant feedback of the ‘push’ model when a server fails to complete all configuration changes correctly, versus waiting for servers to check in on a schedule and getting alerted to the issue some time later.

## Why FreeBSD and NGINX?

FreeBSD has a proven track record for reliability and security. There are numerous options to help secure a FreeBSD system, Jails, separate mount points, capsicum, ZFS, to name but a few. FreeBSD has an excellent security team. Their review is an important piece of the development lifecycle. DTrace is one of the best tools available today. It can be used for everything from troubleshooting and debugging to performance analysis and tuning.

FreeBSD is a fully featured OS with all the applications, tools and documentation necessary to run large-scale enterprise deployments.

The popular streaming provider, Netflix, has built a massive CDN currently responsible for a third of all traffic on the Internet in the US.

The Engineers at Netflix have written extensively on their decisions and highlight FreeBSD’s feature set as the main reasons for their choice of using FreeBSD to build their CDN. Nginx provides the lightweight, tunable web

server needed. The goal of Netflix was summed up by Gleb Smirnov as “get more and more gigabits per second from a single box.” He gave a fantastic presentation on the many decisions that go into choosing and building a large-scale CDN and why FreeBSD and Nginx were the ideal choices.

Although the challenges Netflix faces are vastly different than most companies or individuals face, we can use their work to show how reliable, cost-effective and secure FreeBSD and Nginx can be in large-scale DevOps deployments.

## Ending Goal:

Learning a new configuration management tool can be a challenging task, even more so when the ending goal is undefined. This article will mainly focus on Ansible’s key features and setup, how Ansible fits into the modern DevOps movement and some of the decisions you will make in designing a FreeBSD server infrastructure.

Using FreeBSD as a start, the large number of ports and packages available along with a tool like Ansible enable you to build a large-scale infrastructure quickly.

This infrastructure can grow to nearly any size and span as many data centers or regions as need be. The only limitation I’ve heard of so far is the number of simultaneous SSH sessions the Ansible control server can handle when using Ansible in very large scale environments. Otherwise, the number of nodes you can control with Ansible is endless.

Another benefit of using Ansible with a DevOps approach is that documentation is part of the process. A well-written task will use the name to define the intention of the step; “Install Nginx dependency”, “Configure Application Monitoring agent” or “git clone Web Application XYZ”.

Even if the module being run in the task is too difficult to understand or the code is too complex to read, the name provides good insight into the intention. Thus, it's simple to understand and even beginners can instantly recognize the goal of each task.

### Getting Started:

One of Ansible's best features is that it's client-less. If you have SSH access to the servers, you can begin working with Ansible to deploy software immediately, password-less sudo is helpful but not required. Begin by installing Ansible on your local workstation; 'pip' is currently the preferred method.

```
$ pip install ansible

Collecting ansible

Downloading ansible-2.2.1.0.tar.gz (2.5MB)

100% |#####| 2.5MB
317kB/s

...

...

Installing collected packages: idna, pyasn1, six,
enum34, ipaddress, pycparser, cffi, cryptography,
paramiko, MarkupSafe, jinja2, PyYAML, pycrypto,
ansible

...

...

Running setup.py install for ansible ... done

Successfully installed MarkupSafe-0.23 PyYAML-3.12
ansible-2.2.1.0 cffi-1.9.1 cryptography-1.7.2
enum34-1.1.6 idna-2.2 ipaddress-1.0.18 jinja2-2.8.1
paramiko-

2.1.2 pyasn1-0.2.2 pycparser-2.17 pycrypto-2.6.1
six-1.10.0

$ ansible --version

ansible 2.2.1.0
```

As long as you have Python 2.7 version on the servers, you have everything you need to manage your FreeBSD infrastructure. There's no additional client software to install to manage infrastructure with Ansible.

Additionally, there's no need to create and sign client keys. Ansible does not use certificates like Chef or

Puppet. Instead, it uses SSH hosts keys to verify client identity. If you're working with more than five servers, it may be helpful to use a script that trusts the host keys for you.

Now is a great time to begin thinking about how you'll populate your inventory file(s). You're lucky if you have a virtualized infrastructure with an API to query and get a quick list. An inventory file defines the servers, which groups they belong to and options for each server. Here's an example of an inventory file.

```
$ cat local

[fbs]

fbs-01 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7
fbs-02 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7
fbs-03 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7

[databases]

fbdb-1 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7
fbdb-2 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7

[gitlab]

fbgit1 ansible_ssh_user=freebsd
ansible_python_interpreter=/usr/local/bin/python2.7
```

There's a few options below you may want to add to the Ansible config file that are rather helpful. The 'ssh\_args' improves SSH connection speed (by multiplexing the connections) and key forwarding to simplify the process of pulling project code from your git repo on the target server.

Depending on which local user (or root) is used, you may or may not need to 'sudo' to make system changes. Here's an example 'ansible.cfg' file:

```
$ cat ansible.cfg

[defaults]

become=yes

become_method=sudo

sudo_flags=-HE

[ssh_connection]
```



```
ssh_args = -o ControlMaster=auto -o
ControlPersist=60s -o ForwardAgent=yes
```

Ansible modules are small pieces of code that do specific things on a given system. For example, the ‘user’ module allows you to create a system user or modify a user by changing e.g the shell, editing the password, adding an SSH key, etc. The ‘template’ module allows you to take a Jinja2 template with variables and write out a file. Using variables in the template, entries can change based on environment/location. You can replace the connection strings in the DB config file on each web server, pointing to the closest DB server in the local DC/region. Ansible modules allow you to perform basic tasks on the infrastructure with minimal amount of effort. Let’s test connectivity with Ansible using the ping module:

```
# The command 'ansible' using the 'local'
inventory file, this will

# run on the 'fbs' hosts using the 'ping'
module to test connectivity.
```

```
$ ansible -i local fbs -m ping

fbs-03 | SUCCESS => {

"changed": false, "ping": "pong"

}

fbs-02 | SUCCESS => {

"changed": false, "ping": "pong"

}

fbs-01 | SUCCESS => {

"changed": false, "ping": "pong"

}
```

The significance here is not that the servers responded to an ICMP packet with echo reply, the ping module does much more. Used in this manner, Ansible is making an SSH connection to each host, running a small piece of Python code that responds back with a friendly ‘pong’ message after it completes successfully. This seemingly simple response helps to ensure that you’ve met the base criteria to perform system changes with Ansible.

Many of the Ansible modules run on FreeBSD with no changes. A simple task that creates users can be run on

FreeBSD or Linux with no modification. Depending on your needs the tasks and roles can be very simple or highly complex. I enjoy the polished nature of Ansible. Its default set of modules provides a great deal of functionality.

The Ansible community is alive and active. The best place to look for prebuilt roles is ‘Ansible Galaxy’. You can find roles for all of the popular applications and services; NGINX, Apache, MySQL, PostgreSQL, or even a Jailed Joomla/Wordpress server. Many of the roles available on Ansible Galaxy will help you to build a fully functioning service/application, very little coding is required. Let’s take a look a few pieces of an NGINX role in Ansible Galaxy to dive a little deeper. Head over to GitHub and check out <https://github.com/icamys/ansible-role-nginx>. It’s a role created by “icamys” (Prisacari Dmitrii) with twenty three contributors receiving check-ins on a regular basis. In ‘main.yml’ under tasks, we find the following code.

```
# Nginx setup.

- name: Copy nginx configuration in place.
  template:

    src: nginx.conf.j2

    dest: "{{ nginx_conf_file_path }}" owner: root

    group: "{{ root_group }}" mode: 0644

  notify:

    - reload nginx
```

The template module provides us with a simple way to update the Nginx config file in the above code sample. The last two lines will force the Nginx service to reload if this task made any changes to the config file. Take note of the ‘dest’ and ‘group’ entries. They are variables that can be substituted for many reasons. The Nginx config path could be using a variable so that the file can be placed in the correct location based on the operating system. This particular role is Linux, FreeBSD and OpenBSD compatible. The root group may need to change for the same reason, based on operating system of the target server.

Here are the first few lines of nginx.conf.j2:

```
user  {{ nginx_user }};

error_log  {{ nginx_error_log }}; pid  {{
nginx_pidfile }};

worker_processes  {{ nginx_worker_processes }};
```

The template includes several variables that can be replaced in several ways. You can set defaults that are sane fallback entries. Based on the environment the target server is in, I would frequently set a larger number of worker processes on production systems or change the log file location, perhaps using a mount point only available on production systems.

Here's another example Jinja2 template, the first few lines of `vhosts.j2`:

```
{% for vhost in nginx_vhosts %} server {

listen {{ vhost.listen | default('80
default_server') }};
```

This template allows you to iterate through a list of 'nginx\_vhosts' and creates an entry for every name in the list. This is helpful in ensuring that the template is easy to read, without excessive repetition of config entries, and allows you to create 1 or 100 vhosts entries with ease.

### DevOps KungFu:

It's helpful to group servers based on function. Whether it's 2 or 2,000 instances, it's important to define function as best as possible. Based on the function, certain types of servers should never be accessible to the internet on any level, and many are directly accessible *or indirectly* accessible through a load balancer. Each of the groups will be assigned one or more Ansible roles. The roles will affect which applications are installed, firewall rules applied, users added to the system, mount points

created, etc. Also, the roles can be used to identify the appropriate network interfaces or AWS security groups when making API calls or running scripts to provide the VMs.

Once you have well defined roles, you can use this to produce identical environments. It's rather common to see multiple environments in an applications' lifecycle, most frequently, names like beta, production, staging, etc. Although Docker is heavily publicized today as helping with portability, FreeBSD implemented Jails a very long time ago, and Jails are far more secure. Using Ansible for system configuration offers a much greater flexibility than using a generic container deployment process.

When each environment is configured in exactly the same way, it becomes easier to find bugs earlier in the process. Systems Engineering becomes more about finding anomalies and performance tuning than the constant firefighting that comes from manually configured environments. Development becomes less stressful as bugs and issues can be identified earlier in the process, no need to stay up all night working on a hotfix for a bug live in production. Quality Assurance becomes easier as well. Engineers can be sure that the tests they run in beta or staging will produce the same results as production.

### Flexibility and good app lifecycle payoffs:

In addition to providing identical environments for production, staging, etc. you can also easily move the application to your new virtualized environment from bare metal, or to a brand new data center or the cloud provider of choice. Although most cloud providers offer technology that makes application setup and deployment seem easy using their proprietary software, it greatly limits your choices to switch between the various cloud providers. Flexibility is very important when designing an application infrastructure, being able to move between a local data center and various cloud providers offers freedom and additional High Availability and Disaster Recovery options.

### DevOps role in compliancy

Is it possible to ensure security patches and system updates have been applied to an entire environment without updating each system one by one? This is when a flexible application lifecycle management system really shows how valuable it can be to an entire organization, not just Development and Operations Engineers. Why bother updating each system individually? It takes a great deal of network bandwidth and CPU power to update thousands of nodes, perhaps there's an alternative.

Certain highly secure environments explicitly deny servers the ability to reach out directly to the internet. This may be done to satisfy compliancy requirements or mitigate the risk of data exfiltration. Although it would be trivial (network and CPU resources) to host a local FreeBSD mirror in your DC, it's still an extra cost. Working mainly in AWS environments has forced me to take cost into account in all of my Systems Architecture and Operations decisions. One option is to update one



single host, use the updated host to create a new image, which will be used to rebuild the entire web application infrastructure. This is a particularly helpful option when working in an environment with no internet access.

Provisioning a new set of servers to push out security updates is not likely an effective approach in all situations. In some situations, that approach could be counterproductive; causing congestion on the Storage network, of excessive IO on the SANs. Although there are drawbacks, provisioning a fresh new infrastructure can be used to deal with many types of changes related to compliance. Perhaps, there's a need to swap to a new type of logging solution or monitoring agent. Many issues can arise from updating configuration strings on live running system; locked files, hung processes, services failing to start after applying change or update. Provisioning a fresh new instance with only the correct software installed and configuration settings applied can avoid many different types of issues. Dealing with complicated issues and debugging can be very fun at times. However, avoiding those many different types of issues altogether has its perks as well.

Systems running in an extremely high-secured environment may be configured in ways that greatly limit the ability to make changes to the systems. A final step of an application deployment could be to set certain partitions to mount read-only and increasing the system's security level then rebooting. The running systems would be more difficult for attackers to compromise, but high security environments can create additional hurdles for large scale administration.

So, it's very important to have many approaches and techniques to deal with large scale infrastructure challenges. One of the techniques that is always helpful is making sure that repetitive tasks can be performed using automation, rather than large amounts of manual configuration effort.

## Conclusion

Ansible is a relatively new kid on the block compared to Puppet and Chef. I prefer to work with Python over Ruby, and have found Ansible has reached parity with the functionality of Puppet or Chef. I've had the opportunity to work with nearly every major infrastructure management tool available today. Working with them in large-scale production environments has kept my mind open to the many options available today. I certainly would not say Ansible is better, or Chef is better, or Puppet, but I would say that like all softwares, each has

highlights and drawbacks that must be weighed carefully against each other.

Of all the tools I've used, Chef and Ansible are the top two in my opinion. The only deciding factor in my mind is whether the team supporting it prefers Ruby or Python. Check out Arun Tomar's workshop and Ebook to learn more about DevOps with Chef On FreeBSD.

It's so exciting to see the many options available to manage large-scale FreeBSD server environments. Although Ansible is a relatively new kid on the block, it has all the features necessary to create and maintain secure, efficient and scalable Continuous Integration environments for software deployments.

## References:

Netflix blogs:

<https://www.nginx.com/blog/why-netflix-chose-nginx-as-the-heart-of-its-cdn/> Gleb Smirnov: Why did Netflix use Nginx and FreeBSD to build their own CDN? Ansible

Docs on modules:

[http://docs.ansible.com/ansible/user\\_module.html](http://docs.ansible.com/ansible/user_module.html)

[http://docs.ansible.com/ansible/template\\_module.html](http://docs.ansible.com/ansible/template_module.html)

[http://docs.ansible.com/ansible/list\\_of\\_all\\_modules.html](http://docs.ansible.com/ansible/list_of_all_modules.html)

Ansible Galaxy:

<https://galaxy.ansible.com/intro>

<https://galaxy.ansible.com/list#/roles>

[https://galaxy.ansible.com/list#/roles?page=1&page\\_size=10&autocomplete=freebsd](https://galaxy.ansible.com/list#/roles?page=1&page_size=10&autocomplete=freebsd)

Ansible Galaxy Nginx role:

<https://galaxy.ansible.com/icamys/nginx/>

<https://github.com/icamys/ansible-role-nginx/blob/master/templates/vhosts.j2>

## About the Author

David Rodriguez is Sr. Systems Engineer and Inquisitive Linux Professional focused on large-scale high availability systems using virtualization. Hands-on security engineer at Infection.

# Taking a Look at SmartOS Containers

**SmartOS is an illumos distribution created by Joyent that is aimed at cloud computing the main business of Joyent. SmartOS acts as a hypervisor where you can create VMs using KVM which was ported by SmartOS team from Linux to SmartOS or bare metal containers/zones that will host illumos guests or even Linux ones.**

But what is illumos? Illumos is the open-source fork of the former OpenSolaris project which started at Sun Microsystems. The former engineers that worked on Solaris/OpenSolaris made it available for the open-source community. Here are some of the features that SmartOS/illumos brings to the table, taken from the SmartOS wiki page

## **Operating System Virtualization**

*"Thanks to the Solaris/Illumos heritage, SmartOS already had Containers and Zones -- container-based virtualization (containers is supposed to mean zones + resource controls) that allowed users to run multiple applications set on one server isolated from one another. With KVM on SmartOS, Joyent can now address workloads that require running a full operating system for those customers who need Linux, Windows or other OS to run in full, hardware-assisted virtualization. Unlike any other "hypervisor", Joyent's KVM images run as a process inside of a zone. It turns out to be a very secure way to run Windows. And, unlike Linux, SmartOS gives customers access to Solaris technologies that many users find compelling -- like DTrace and ZFS. "*

This is proven technology which dates back from the Solaris ten days. It has been battle tested even the lx zones ran great back in the day. Now the lx brand (a brand is a concept to name which OS runs in a zone. In this case, lx means Linux) has been resurrected, updated and works great!

## **Resource Controls**

SmartOS offers two methods for controlling CPU consumption:

Fair share scheduler lets the operator set a *minimum* guaranteed share of CPU. It takes effect when the system is busy due to the demand from more than one zone, to ensure that each gets its fair share. Otherwise, when the system is not busy, a zone can "burst" beyond its usual limit, consuming more than the minimum as guaranteed, up to the CPU cap set for it.

CPU cap is a *maximum*, e.g. an amount of CPU time that a user has paid for. This can also be used to set user expectations about system performance, even when the overall system is not yet populated and the workload is still light.

## **Network Virtualization**

Virtualization is also used to create the illusion of things that aren't actually on the real system, such as virtual network interfaces (VNICs). Joyent was one of the first users of Project Crossbow, which added network virtualization to OpenSolaris. Using this technology, each Joyent SmartMachine gets up to 32 VNICs, each with its TCP/IP stack. This helps to maximize another scarce resource, IPv4 addresses, through the use of network pools.

## **Observability**

Users of Illumos, Mac OS X and FreeBSD know that DTrace gives them an unprecedented view of what's going on throughout the software stack. In SmartOS, it allows operators to observe and troubleshoot across all the zones and nodes in an entire data center. In SmartDataCenter, the Joyent team has harnessed the power of DTrace in a more user-friendly form with Cloud



Analytics, which is available to both cloud operators and their customers.

## Security

Solaris has long been the operating system of choice in highly secure data centers, thanks to several features which SmartOS inherits. SmartOS zones, though they share system resources such as CPU and disk space, simply cannot see each other. Users in a multi-tenant environment are thus protected from each other i.e. your neighbor's security lapse will not affect your zone. Additionally, data security is also assured. No byte of data from one customer is shared with any other customer, now or later, because:

A zone can only see its own network traffic.

Disk storage is accessed only via ZFS file systems, never raw devices. Each SmartMachine has its own file system and does not even know of the *existence* of any other.

A user has no access to raw memory devices. So, he/she can't scan the system's memory.

Upon deletion of a SmartMachine, the file system is destroyed and there is no device path by which a future customer could access any data left over in that file system. A SmartMachine is protected from DDOS attacks by some of the same features that guarantee that it gets a fair share of system resources: fair share scheduler, caps, process limits, rcapd, swap cap, disk file system limits, quota limits.

By capping each zone's resource usage, SmartOS ensures that, even under heavy attack, a zone will not bring down its neighbors.

## ZFS

If you don't know ZFS, you should start learning about it right now. Here is a couple of features that come with ZFS, the last word on file systems

Data integrity is guaranteed, with particular emphasis on preventing silent data corruption.

Storage pools: "virtualized storage" makes administrative tasks and scaling far easier. To expand storage capacity, all you need to do is add new disks (hard disks, flash memory and whatever may come along in the future) to a zpool.

Snapshots: ZFS' copy-on-write transactional model makes it possible to capture a snapshot of an entire file system at any time, storing only the differences between that and the working file system as it continues to change. This creates a backup point that the administrator can easily roll back to.

Clones: Snapshots of volumes and file systems can be cloned, creating an identical copy. Cloning is nearly instantaneous and initially consumes no additional disk space. This facilitates the rapid creation of new, nearly identical, VMs.

\* The ARC (Adaptive Replacement Cache) improves the file system and disk performance, driving down overall system latency.

## Scalability

This is an enterprise battle tested OS that had been the choice for enterprise computing for several years!

## Reliability

Fault management (FMA): "fine-grained fault isolation and restart where possible of any component — hardware or software — that experiences a problem. To do so, the system must include intelligent, automated, proactive diagnoses of errors that are observed on the system. The diagnosis system is used to trigger targeted automated responses or guided human intervention that mitigates a specific problem or at least prevents it from getting worse."

The Service Management Facility (SMF) is "a feature of the Solaris operating system that creates a supported, unified model for services and service management on each Solaris system".

## Installing SmartOS

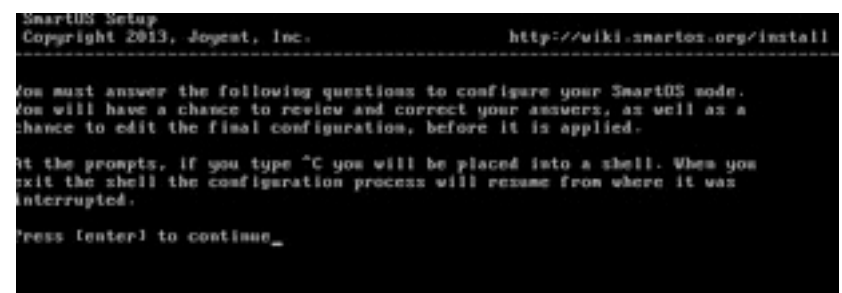
If you are going to install into a bare metal, it is a good idea to check first the Illumos hardware compatibility list, and also the hardware requirements for SmartOS.

<https://wiki.smartos.org/display/DOC/Hardware+Requirement+S>

Download the usb image, vmware image or iso from

<https://wiki.smartos.org/display/DOC/Download+SmartOS>

Now that we have the SmartOS image, let's start installing it. SmartOS runs as a hypervisor. Therefore, it runs only in memory so that the space allocated for a root dataset is not wasted. We will only focus in our containers.



```

SmartOS Setup
Copyright 2013, Joyent, Inc.      http://wiki.smartos.org/install

You must answer the following questions to configure your SmartOS node.
You will have a chance to review and correct your answers, as well as a
chance to edit the final configuration, before it is applied.

At the prompts, if you type ^C you will be placed into a shell. When you
exit the shell the configuration process will resume from where it was
interrupted.

Press [enter] to continue_

```

```
SmartOS Setup
Networking                                     http://wiki.smartos.org/install

To set up networking you must first configure a network tag. A network tag
refers to a physical NIC or an aggregation. Virtual machines will be created on
top of a network tag. Setup will first create a network tag and configure a NIC
so that you can access the SmartOS global zone. After setup has been completed,
you will have the option of creating additional network tags and configuring
additional NICs for accessing the global zone through the nictagadm(1M) command.

Press (enter) to continue_
```

```
SmartOS Setup
Storage                                         http://wiki.smartos.org/install

SmartOS will automatically determine what we think is
the best zpool layout from your current disks. You may use this
suggestion, change to another built in storage profile, or simply create
your own zpool.
udev: c2t0d0 total capacity: 50.00 GB

This is the 'default' storage configuration. To use it, type 'yes'.
To see a different configuration, type: 'raidz2', 'mirror', or 'default'.
To specify a manual configuration, type: 'manual'.

_
```

I'll choose default. If you have more than one disk, you could use raidz2, mirror or choose manual to create zlogs, cache and whatever you need.

```
SmartOS Setup
Networking ~ Admin                             http://wiki.smartos.org/install

The admin network is the primary network in SmartOS. It is the default network
that is created. The configured NIC will be used to access the global zone. If
you wish to use a VLAN on this network, you must configure VLAN ACCESS mode for
this network.

Number Link      MAC Address      State  Network
1       e1000g0  00:0c:29:b4:19:b9  up    -
2       e1000g1  00:0c:29:b4:19:c3  up    -
Enter the number of the NIC for the 'admin' interface: _
```

Here, all our network interfaces will appear. We need to choose one of them to configure it as an administrator interface. Thereafter, we could set up the rest of the interfaces.

At this time, we will only use the admin interface.

Press enter then assign an IP address or just type dhcp.

```
SmartOS Setup
System Configuration                           http://wiki.smartos.org/install

Setup will now go through and prompt for final pieces of account configurations.
This includes setting the root password for the global zone and optionally
setting a hostname.

Enter root password: _
```

```
SmartOS Setup
System Configuration                           http://wiki.smartos.org/install

Setup will now go through and prompt for final pieces of account configuration.
This includes setting the root password for the global zone and optionally
setting a hostname.

Enter root password:
Confirm password:
The entries do not match, please re-enter.
Enter root password:
Confirm password:
Enter system hostname (press enter for none): s01_
```

```
SmartOS Setup
Networking ~ Continued                         http://wiki.smartos.org/install

The default gateway will determine which router will be used to connect the
global zone to other networks. This will almost certainly be the router
connected to your 'admin' network. Use 'none' if you have no gateway.

Enter the default gateway IP (none): _
```

In my case, my IP is 192.168.1.1

```
SmartOS Setup
Networking ~ Continued                         http://wiki.smartos.org/install

The default gateway will determine which router will be used to connect the
global zone to other networks. This will almost certainly be the router
connected to your 'admin' network. Use 'none' if you have no gateway.

Enter the default gateway IP (none): 192.168.1.1

The DNS servers set here will be used to provide name resolution abilities to
the SmartOS global zone itself. These DNS servers are independent of anything
you use to create virtual machines through vmadm(1M).

Enter the Primary DNS server IP (8.8.8.8): _
```

```
SmartOS Setup
Verify Configuration                           http://wiki.smartos.org/install

Please verify your SmartOS Configuration. After this point the system will set
up and all data on the disks will be erased.

Net      MAC      IP addr.  Netmask
Admin 00:0c:29:b4:19:b9  dhcp      N/A

DNS Servers: (192.168.1.1, 8.8.4.4), Search Domain: org
Hostname: s01
NTP server: 0.smartos.pool.ntp.org

Is this correct, proceed with installation? (y): _
```

Also here, my IP is 192.168.1.1



```

Please verify your SmartOS Configuration. After this point the system will set
up and all data on the disks will be erased.

Net      MAC      IP addr.  Netmask
Admin 00:0c:29:b2:19:b9      dhcp      N/A

DNS Servers: (192.168.1.1, 8.8.4.4), Search Domain: org
Hostname: s01
NTP server: 0.smartos.pool.ntp.org

Is this correct, proceed with installation? (y): y
Creating pool zones... done
Taking dump zvol... done
Initializing config dataset for zones... done
Creating config dataset... done
Creating global cores dataset... done
Creating opt dataset... done
Initializing var dataset... done
Creating swap zvol... done
System setup has completed.
Press enter to reboot.

```

If all goes well, you will see this login screen



## SmartOS zones/containers

Containers were inspired by FreeBSD jails. In Solaris 10, they are called zones. A zone is a virtualized instance that behaves like an isolated system even when functioning alongside other zones on the same machine. Each zone on a system shares a pool of resources and the single operating system kernel. However, zones are never aware of other zones on the system and are process secure. A zone is similar to a virtual machine, but is distinct in that it shares the base system kernel whereas each virtual machine runs its own OS kernel. Zones are an inherent part of the operating system and impose no additional overhead. Each process that runs includes the zone ID as an attribute. Thus, zones scale and perform better than virtual machines since there is no additional kernel or layering involved.

## Creating SmartOS containers

To create containers, we use the handy *vmadm* tool that comes with SmartOS.

*vmadm* allows you to interact with virtual machines on a SmartOS system.

At the moment, there are three types of machines:

- OS Virtual Machines (SmartOS zones).
- LX Brand (Linux zones).
- KVM Virtual Machines.

All of them can be managed with *vmadm*. It allows you to create, inspect, modify and delete virtual machines on the local system.

The steps to create a container are as follows:

- Download an image that the container will be based on. This done using the *imgadm* command.
- Create a json file to describe your container/vm configuration.
- Create the container using the *vmadm create* command and the json file previously created.

This tool allows you to import and manage virtual machine images on a SmartOS system. Virtual machine images (also sometimes referred to as 'datasets') are snapshots of pre-installed virtual machines which are prepared for generic and repeated deployments.

First, let's add Joyent to our source of images:

**\$ *imgadm sources -a https://images.joyent.com***

Images are identified by UUID. So, let's fetch this one, why this one? Because it's the recommended image to build SmartOS live. You could choose another one.

**\$ *imgadm import e69a0918-055d-11e5-8912-e3ceb6df4cf8***

You could check the ones available using:

**\$ *imgadm avail***

Now that we have an image, we are able to start creating our container/zone/vm, whatever name you choose; it's described at creation by a json file. Here is an example for a SmartOS based container.

Save this file as *build01.json* or whatever name you choose.

```
{
  "brand": "joyent",
  "fs_allowed": "ufs,pcfs,tmpfs",
  "image_uuid": "e69a0918-055d-11e5-8912-e3ceb6df4cf8",
  "alias": "build01",
  "hostname": "b01",
  "max_physical_memory": 2024,
  "quota": 10,
  "resolvers": ["8.8.8.8", "192.168.1.1"],
  "nics": [
    {
      "nic_tag": "admin",
      "ips": ["dhcp"],
      "primary": true
    }
  ]
}
```

I'll describe each of the keys used. There are more but for this example, we will only use these ones. The rest of the keys are described in the vmadm

**brand:** This will be one of 'joyent', 'joyent-minimal' or 'lx' for OS virtualization and 'kvm' for full hardware virtualization. This is a required value for VM creation.

**image\_uuid:** The UUID of the image you are using as a template in our case is the one we just fetched

**alias:** An alias for a VM which is for display/lookup purposes only. Not required to be unique.

**fs\_allowed:** This option allows you to specify file system types this zone is allowed to mount. As I use this

container to build SmartOS live, the values used are the recommended ones.

**resolvers:** For OS VMs (not KVM ones), this value sets the resolvers which is placed in /etc/resolv.conf at VM creation. If maintain\_resolvers is set to true, updating this property will also update the resolvers in /etc/resolv.conf. For KVM VMs, these will get passed as the resolvers with DHCP responses.

**nics:** When creating a KVM VM or getting a KVM VM's JSON, you will use this property. This is an array of 'nic' objects. The properties available are listed below under the nics.\*.<property> options. If you want to update nics, see the special notes in the section above about the 'upgrade' command. When adding or removing NICs, the NIC names will be created in the order of the interfaces in the nics or add\_nics array.

**quota:** This sets a quota on the zone file system. For OS VMs, this value is the the quota for the Zone containing the VM, which is not directly Set quota to 0 to disable (i.e. for no quota).

**nics.\*.ips:** An array of IPv4 or IPv6 addresses to assign to this NIC. The addresses should specify their routing prefix in CIDR notation. The strings 'dhcp' (DHCPv4) and 'addrconf' (SLAAC or DHCPv6) can also be used to obtain the address dynamically. Up to twenty addresses can be listed.

**nics.\*.nic\_tag:** This option for a NIC determines which host NIC, the VMs nic, will get attached to. The value can

```
root@b01 ~]# pkgin install apache
calculating dependencies... done.

nothing to upgrade.
packages to be installed (8101K to download, 29M to install):

pr-util-1.5.4nb1 apr-1.5.1 apache-2.4.23nb2

proceed ? [Y/n] y
downloading packages...
pr-util-1.5.4nb1.tgz          100% 404KB 134.6KB/s 195.9KB/s   00:03
pr-1.5.1.tgz                 100% 677KB 225.6KB/s 203.8KB/s   00:03
apache-2.4.23nb2.tgz         100% 7020KB 334.3KB/s 215.0KB/s   00:21
installing packages...
installing apr-util-1.5.4nb1...
installing apr-1.5.1...
installing apache-2.4.23nb2...
apache-2.4.23nb2: Creating group ``www''
apache-2.4.23nb2: Creating user ``www''
passwd: password information changed for www
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-autoindex.conf to /opt/local/etc/httpd/httpd-autoindex.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-dav.conf to /opt/local/etc/httpd/httpd-dav.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-default.conf to /opt/local/etc/httpd/httpd-default.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-info.conf to /opt/local/etc/httpd/httpd-info.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-languages.conf to /opt/local/etc/httpd/httpd-languages.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-manual.conf to /opt/local/etc/httpd/httpd-manual.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-mpm.conf to /opt/local/etc/httpd/httpd-mpm.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-multilang-errordoc.conf to /opt/local/etc/httpd/httpd-multilang-errordoc.conf
apache-2.4.23nb2: copying /opt/local/share/examples/httpd/extra/httpd-ssl.conf to /opt/local/etc/httpd/httpd-ssl.conf
```



either a nic tag as listed in the 'NIC Names' field in 'sysinfo' or an etherstub or device name.

**nics.\*.primary:** This option selects which NIC's default gateway and name server values will be used for this VM. If a VM has any nics, there must always be a one primary. Setting a new primary will unset the old. Trying to set two nics to primary is an error.

**max\_physical\_memory:** The maximum amount of memory on the host that the VM is allowed to use.

Before creating the container, first validate the VM description using vmadm, just type:

**\$ vmadm validate -f build01.json**

It will error out if there is a mistake in the json file, and in what key/value it is.

Using this json file, the output should be:

**\$ VALID 'create' payload for joyent brand VMs.**

Now, let's create it

**\$ vmadm create -f build01.json**

If the container is created successfully, it should be listed on the containers lists

**\$ vmadm list**

UUID	TYPE	RAM	STATE	ALIAS
------	------	-----	-------	-------

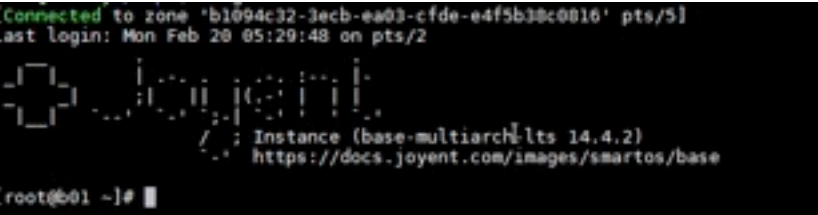
8bd4af70-5751-cc85-ad93-c41dd9c797e5	LX	512	running	lxtest
--------------------------------------	----	-----	---------	--------

b1094c32-3ecb-ea03-cfde-e4f5b38c0816	OS	2024	running	build01
--------------------------------------	----	------	---------	---------

Now, login to your zone/container using UUID

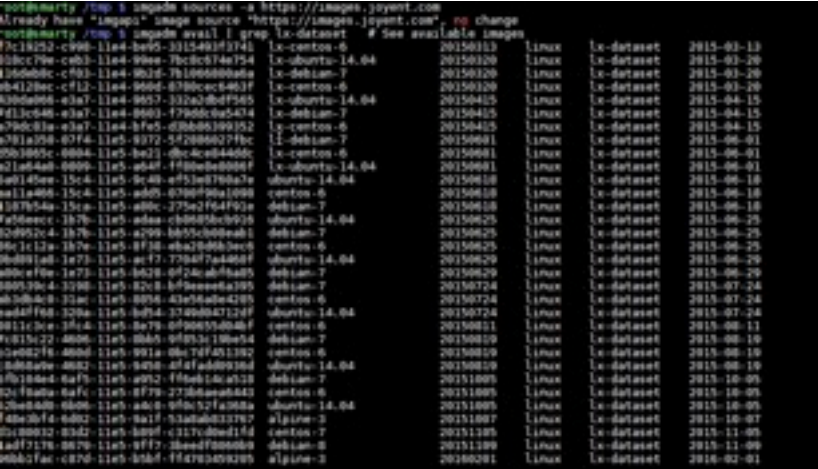
**\$ zlogin b1094c32-3ecb-ea03-cfde-e4f5b38c0816**

And you will see this:



SmartOS uses pkgsrc as a package manager. So in our newly created container, we could install whatever package is in the joyent repo <https://pkgsrc.joyent.com/>. For example, let's install the old trusty apache webserver.

**\$ pkgin install apache**



Creating Linux containers (lx brand zones)

If you are suffering a vendor lock-in, and your application only runs in Linux as you don't have the source code, or you have the source code and it's full of *linuxisms* and don't have the time to port it or just simply are most used to Linux, then lx zones could save the day.

You could run your Linux application unmodified just as you usually do, but will enjoy all the goodies from Illumos/SmartOS like ZFS and Dtrace, which is amazing.

As we did with the SmartOS container, we need a vm image, let's fetch one.

Add Joyent image sources (because you could also host your own images) and then check which Linux containers are available.

**\$ imgadm sources -a https://images.joyent.com**

**\$ imgadm avail | grep lx-dataset**

Let's create an Ubuntu zone as illustrated in the SmartOS documentation,

**\$ imgadm import 05140a7e-279f-11e6-aedf-47d4f69d2887**

As we did for the Illumos zone, we need to create a json file that will describe the VM and use the image we just downloaded

Save this as lx.json or whatever name you choose.

```
{
  "alias": "lptest",
  "brand": "lx",
  "kernel_version": "4.3.0",
  "max_physical_memory": 512,
  "quota": 10,
  "image_uuid": "05140a7e-279f-11e6-aedf-47d4f69d2887",
  "resolvers": ["8.8.8.8", "192.168.1.1"],
  "nics": [
    {
      "nic_tag": "admin",
      "ips": ["dhcp"],
      "primary": true
    }
  ]
}
```

The only keys that are new in this json are the following:

**kernel\_version:** This sets the version of Linux to emulate for LX VMs.

**brand:** lx because this is a Linux zone

As usual before creating the VM, validate the json file using vmadm

**\$ vmadm validate create -f lx.json**

Login into the new lx zone with zlogin, use the container UUID.

```
oot@lptest: /tmp $ vmadm list
UUID                                TYPE  RAM   STATE  ALIAS
2b44af70-5751-cc85-ad93-c41dd9c797e5 LX    512   running lptest
1094c32-3ecb-ea03-cfde-e475b3b0885b OS    2048   running build001
oot@lptest: /tmp $ zlogin 2b44af70-5751-cc85-ad93-c41dd9c797e5
Connected to zone '2b44af70-5751-cc85-ad93-c41dd9c797e5' pts/5
apt login: Mon Feb 20 03:31:21 UTC 2017 from zone:global on pts/4
welcome to Ubuntu 16.04 LTS (GNU/Linux 4.3.0 x86_64)

* Documentation: https://help.ubuntu.com/

+-----+
|               |
|   joyent      |
|               |
+-----+
Instance (Ubuntu 16.04 20160601)
https://docs.joyent.com/images/container-native-linux
oot@2b44af70-5751-cc85-ad93-c41dd9c797e5:~#
```

The first thing we need to do before we start working on this zone is to create users. As usual, type the following commands:

**\$ adduser -d /home/<youruserhomedir> <username>**

**\$ mkdir /home/<your\_user\_home\_dir> && chown -R <username> /home/<your\_user\_home\_dir>**

Set the password for this user:

**\$ passwd <username>**

Set the root password since it's not setup by default in this container

**\$ passwd**

By default, we cannot login into this zone using ssh. Hitherto, we just need to modify /etc/ssh/sshd\_config and change the following from no to yes:

**# Change to no to disable tunnelled clear text passwords**

**PasswordAuthentication yes**

Also, add the following as we want to forward X11 applications:

**X11Forwarding yes**

**X11UseLocalhost no**

Now restart the sshd server:

**\$ service sshd restart**

Then, let's run apt-get

**\$ apt-get update**

```
oot@2b44af70-5751-cc85-ad93-c41dd9c797e5:~# apt-get update
apt login: Mon Feb 20 03:31:21 UTC 2017 from zone:global on pts/4
welcome to Ubuntu 16.04 LTS (GNU/Linux 4.3.0 x86_64)

* Documentation: https://help.ubuntu.com/

+-----+
|               |
|   joyent      |
|               |
+-----+
Instance (Ubuntu 16.04 20160601)
https://docs.joyent.com/images/container-native-linux
oot@2b44af70-5751-cc85-ad93-c41dd9c797e5:~# apt-get update
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Hit:4 http://security.ubuntu.com/ubuntu xenial-security/main Sources [60.7 kB]
Hit:5 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [219 kB]
Hit:6 http://archive.ubuntu.com/ubuntu xenial-updates/main Sources [232 kB]
Hit:7 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [78.6 kB]
Hit:8 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [478 kB]
Hit:9 http://archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [483 kB]
Fetched 1,674 kB in 8s (209 kB/s)
Reading package lists... 30%
```

All is working perfectly! Remember that we are using an Illumos kernel with the Linux personality. This means that Linux syscalls are translated into Illumos syscalls. Therefore, there is almost no overhead in running an lx zone instead of a full machine virtualization with KVM.

At this time, let's try a X11 application, let's install Firefox.

**\$ apt-get install firefox install libglu1-mesa**

Firefox needs libGL.so. Thus, we will install it. But when we try to execute Firefox, we'll get this error:



**\$ Couldn't open libGL.so.1: libGL.so.1: cannot open shared object file: No such file or directory**

Let's check our IP that connects to this lx zone using ssh and test port forwarding

**\$ ifconfig -a**

```
root@b4d4ef70-5751-cc85-ad93-c41dd9c797e5:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 22:31:dc:40:77:41
          [    ] inet addr:192.168.44.130  Bcast:192.168.44.255  Mask:255.255.255.0
          [    ] inet6 addr: fe80::2031:dfff:fe40:7741/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:262712 errors:0 dropped:0 overruns:0 frame:0
          TX packets:279902 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:149792093 (149.7 MB)  TX bytes:165408400 (165.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MULTICAST  MTU:8232  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@b4d4ef70-5751-cc85-ad93-c41dd9c797e5:~#
```

**\$ ssh -XI <username> <host>**

Now that we are in, let's execute Firefox by typing:

**\$ firefox**

It should appear on your screen if X11 forwarding is working on your side. Extensions not working as a Linux syscall called splice

<http://man7.org/linux/man-pages/man2/splice.2.html>

must be implemented in the lxbrand syscall table, but this is no show stopper.

## Conclusion

There are a lot of neat things to do with SmartOS as a home server. For instance, run containers for development or production in your company, for speed up development using containers, game servers etc. I started knowing about zones in Solaris 10 from running several Linux game servers which run great under the lx brand. I could debug and tune performance using Dtrace, now is even better.

SmartOS is built on the legacy of OpenSolaris, now Illumos. Moreover, there are top talents working on this. Joyent hired all the big guns to work on SmartOS and the result is just amazing. If you like the way containers are handled or if you like docker, just check Triton from Joyent. I'll talk about Triton in other issue.

## References

Here is a couple of links for you to check more on SmartOS:

### Joyent's github repository

<https://github.com/joyent>

### Triton

<https://github.com/joyent/triton>

### SmartOS wiki

<https://wiki.smartos.org/display/DOC/LX+Branded+Zones>

### Why SmartOS?

<https://wiki.smartos.org/display/DOC/Why+SmartOS+-+ZFS%2C+KVM%2C+DTrace%2C+Zones+and+More>

### Avoiding *Linuxisms*

<https://wiki.freebsd.org/AvoidingLinuxisms>

## About the Author

Carlos Antonio Neira Bustos has worked several years as a developer using C, C++, as a kernel developer and debugging enterprise legacy applications. He is currently employed as a C developer under Z/OS, debugging and troubleshooting legacy applications for a global financial company. In his freetime he contributes to opensource projects.

# OpenBSD as a Gateway Firewall for SOHO and Enterprise Networks

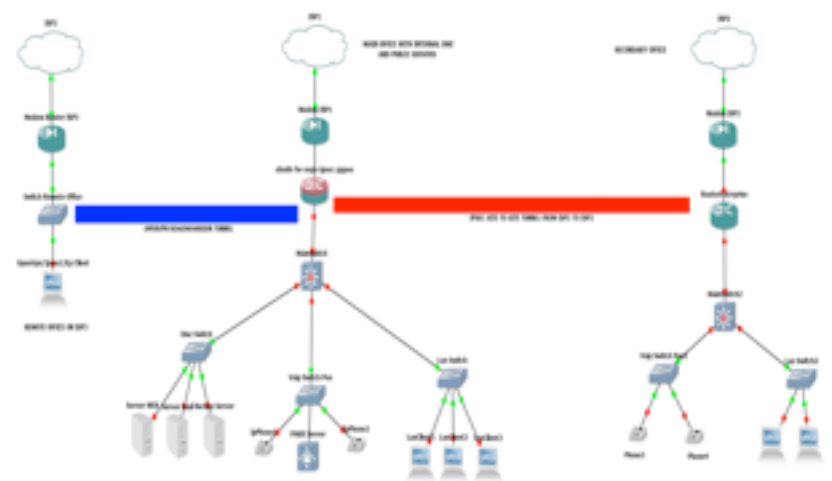
OpenBSD is the second most used BSD operating system with a percentage standing at around 33%, preceded by FreeBSD (70%) and followed by NetBSD and DragonflyBSD. The project slogan "Only two remote holes in the default install in more than ten years."

It is widely used in various enterprise fields and not for the creation of Web servers, mail servers, DNS and network firewall configuration through the powerful PF made available free of charge with the basic installation.

## Creating a Router / Firewall with OpenBSD

A router is a network device, a computer network packet switching which takes care of routing data. It is subdivided into packets between different subnets. Therefore, a logical level, an internal node of the network to the deputy level switch three of the OSI model. The routing may be either to subnetworks connected directly and on separate physical interfaces or to other subnets neighboring, thanks to the information contained in the routing tables. A key feature of the router is to use IP addresses to level the stackTCP / IP compared to switches that route at the local level based on the layer two addresses.

The elements of the routing table may correspond both to individual computers, both to entire networks (SubNet\_ID) or subsets, also very large address space. This is critical to the scalability of the networks as it allows to handle even very large networks by growing the routing tables in a less than linear with respect to the number of hosts.



A firewall is a software or hardware resource that checks information coming from the Internet or another network type, blocking or allowing access to your computer, depending on the security policy set. Its main function is to act as filters for controlling all the network traffic that comes from outside, as well as what is generated from the inside. It allows only that traffic which is authorized. The primary need of every modern company is to define policies to secure access to a priori network. Perimeter security requires optimal and high-quality solutions that guarantee and adequately control the traffic generated.

## I file sysctl.conf, rc.conf ed rc.conf.local

`#/etc/sysctl.conf`

```
net.inet.ip.forwarding=1          # (default:0)
Enables IP FORWARDING

net.inet.ah.enable=1              # (default:1)
Enables AH

net.inet.esp.enable=1             # (default:1)
Enables ESP

net.inet.esp.udpenetcap=1         # (default:1)
Enables UDP encapsulation
```



```

net.inet.esp.udpcap_port=4500      # (default:4500)          else
Sets port for UDP encapsulation

net.inet.ipcomp.enable=1           # (default:0)          eval `/usr/bin/tset -IsQ
Enables IPsec compression         '-munknown:?vt220' $TERM`

net.pipex.enable=1                 # (default:0)          fi
Enables PIPEX                     fi

net.inet.gre.allow=1               # (default:0)          ;;
Enables GRE                       esac

```

### **#/etc/rc.conf.local**

```
pkg_scripts="dnsmasq"
```

### **#/etc/rc.conf**

```
isakmpd_flags="-K"
```

```
npppd_flags=""
```

```
pf=YES          # Packet filter / NAT
```

```
ipsec="YES"      # IPsec
```

### **Profile for user root**

```
# $OpenBSD: dot.profile,v 1.9 2010/12/13 12:54:31 millert
Exp $
```

```
#
```

```
# sh/ksh initialization
```

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/sbin:/usr/local/bin
```

```
export PATH
```

```
: ${HOME="/root"}
```

```
export HOME
```

```
export
```

```
PKG_PATH=http://openbsd.mirror.garr.it/pub/OpenBSD/6.0/packages/i386/
```

```
PS1="\u@\h[\w]"
```

```
export PS1
```

```
umask 022
```

```
case "$-" in
```

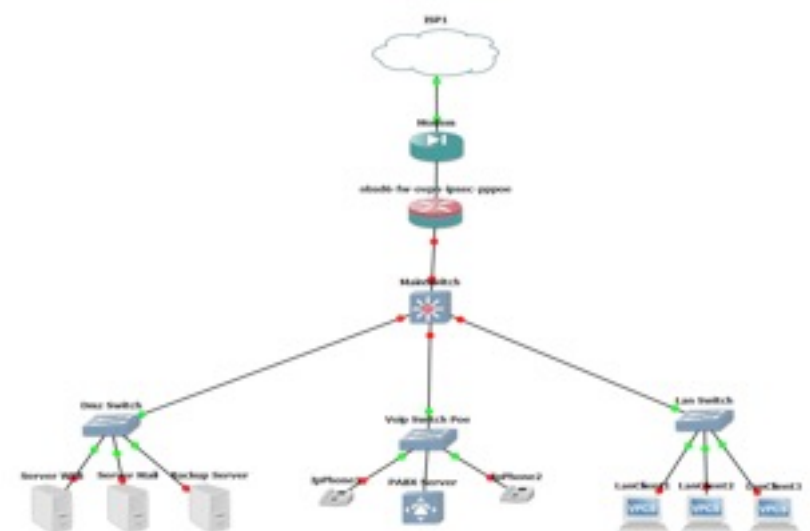
```
*i*)      # interactive shell
```

```
    if [ -x /usr/bin/tset ]; then
```

```
        if [ X"$XTERM_VERSION" = X"" ]; then
```

```
            eval `/usr/bin/tset -sQ
```

```
'-munknown:?vt220' $TERM`
```



## **VLAN: how to configure them and why**

VLANs are an easy, fast and secure means to segment our network and securing the devices that most interest us without having to use different IP addresses. VLANs are managed at level 2 of the ISO / OSI stack or TCP / IP. As a result, the main configuration will be made right in the network switches that will be connected to the devices.

For example, the devices that will connect to ports 1 - 2 - 3, can only exchange data between them and will not see, in any case, other devices connected to the same switch or other. Conversely, applies the same theory.

A PC connected to port 7 of the switch will not know of the existence of a server that is connected to the second because they belong to two different VLAN. Using the VLAN is an excellent practice to defend against network issues, virus attacks, misconfigurations endpoint and especially to decide beforehand who can talk to whom without configuring a different IP network. Take the case

of a company that has twenty-five computer stations, three network printers, three access points and two cameras.

Using VLAN allows us to divide the network on the basis of such offices. The administration office will have its access point and its printer and can talk and see only their PC.

If within the dedicated VLAN a computer is infected with a virus, the virus can only cause damage to members of the VLAN. This is because, all the other PCs on the network will not be visible immediately and therefore immune to any attack. The same company also offers free Wi-Fi to its guests. Upon configuring a dedicated VLAN on the access point, those who will connect to a specific SSID will be automatically tagged and will only see what we decided on the priori network. Quite often, a Guest can access only sail to internet, even if they try to browse the internal network, they would not see anything.

**Gateway and dns settings for our OpenBSD 6 Router Firewall:**

WAN interface ( /etc/hostname.em0 ):

```
inet 192.168.44.40 255.255.255.0 NONE

    inet alias 192.168.44.41 255.255.255.0

    inet alias 192.168.44.42 255.255.255.0

#!route add -net 192.168.202.0/24 192.168.44.50

#!route add -net 192.168.203.0/24 192.168.44.50
```

Router firewall hostname ( /etc/myname ):

```
obsd6-fwgw.domain.tld
```

Router firewall gateway ( /etc/mygate ):

```
192.168.44.2
```

Router firewall Dns settings ( /etc/resolv.conf ):

```
search domain.tld

nameserver 192.168.10.1

nameserver 192.168.10.1
```

```
nameserver 192.168.10.1
```

```
nameserver 8.8.8.8
```

```
lookup file bind
```

Physical LAN interface ( /etc/hostname.em1 ):

```
up # Vlan hardware interface
```

Physical MANAGEMENT interface ( /etc/hostname.em2 ):

```
inet 192.168.201.1 255.255.255.0 NONE
```

**Vlan Setup**

VLAN interface for LAN CLIENTS ( /etc/hostname.vlan10 ):

```
inet 192.168.10.1 255.255.255.0 NONE vlan 10 vlandev em1
```

VLAN interface for DMZ SERVERS ( /etc/hostname.vlan20 ):

```
inet 192.168.20.1 255.255.255.0 NONE vlan 20 vlandev em1
```

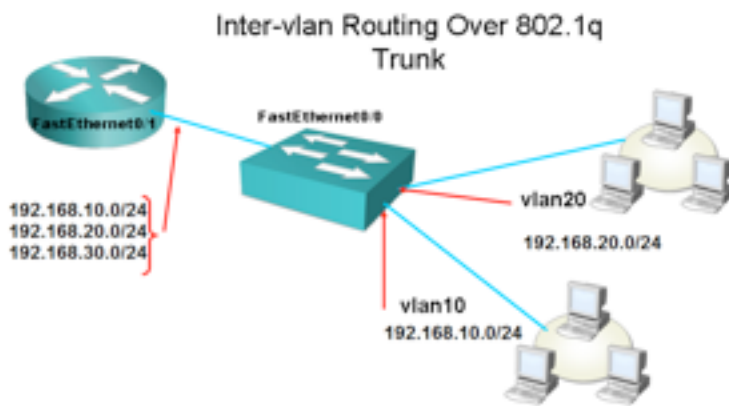
VLAN interface for VOIP CLIENTS ( /etc/hostname.vlan30 ):

```
inet 192.168.30.1 255.255.255.0 NONE vlan 10 vlandev em1
```

**DHCP DNS**

Dnsmasq is a lightweight DNS, TFTP, PXE, router advertisement and DHCP server. It is intended to provide a coupled DNS and a DHCP service to a LAN. It accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. Additionally, it loads the contents of /etc/hosts so that local hostnames which do not appear in the global DNS can be resolved. Also, it answers DNS queries for DHCP configured hosts and can also act as the authoritative DNS server for one or more domains, allowing local names to appear in the global DNS.





The dnsmasq DHCP server supports static address assignments and multiple networks. It automatically sends a sensible default set of DHCP options, and can be configured to send any desired set of DHCP options, including vendor-encapsulated options. It runs also as a secure, read-only, TFTP server to allow net/PXE boot of DHCP hosts and also supports BOOTP. The server includes a proxy mode which supplies PXE information to clients whilst DHCP address allocation is done by another server.

### Configuration files for dns, dhcp and vlan

Main configuration file for dnsmasq a DHCP/DNS server ( /etc/dnsmasq.conf ):

```
# Configuration file for dnsmasq OpenBsd Vlan Server.

#no-resolv

no-poll

interface=lo0

no-dhcp-interface=lo

conf-dir=/etc/dnsmasq.d

resolv-file=/etc/dnsmasq.conf.resolv

# Authoritative DHCP SERVER

dhcp-leasefile=/var/log/dnsmasq.leases

dhcp-authoritative

# Debugging DNS queries.

log-queries

# Logging DHCP transactions.

log-facility = /var/log/dnsmasq.log

log-dhcp
```

```
# DNSSEC setup
```

```
#dnssec
```

```
#trust-anchor=.,19036,8,2,49AAC11D7B6F6446702E54A16073716
07A1A41855200FD2CE1CDDE32F24E8FB5
```

```
#dnssec-check-unsigned
```

Main configuration file for dnsmasq cache resolver ( /etc/dnsmasq.conf.resolv):

```
nameserver 208.67.220.220
```

```
nameserver 8.8.8.8
```

```
nameserver 8.8.4.4
```

Specific configuration files for any vlan are stored in /etc/dnsmasq.d/vlanX.conf file:

Configuration file for LAN VLAN:

```
# Configuration file for dnsmasq OpenBsd Vlan Server.

interface=vlan10

listen-address=192.168.10.1

# Insert domain after hostname

domain=domain10.tld

expand-hosts

# Setting DHCP Ranges

dhcp-range=vlan10,192.168.10.20,192.168.10.200,255.255.25
5.0,48h

# Static DHCP Settings lan 10

#dhcp-host=00:0F:FE:33:CD:AE,192.168.10.3 # Asterisk
Server

#dhcp-host=00:48:54:5B:F8:EB,192.168.10.4 # VÅmware
Server

# SAMBA/WINBINDD Dhcp Server Options

dhcp-option=19,0 # option ip-forwarding off

dhcp-option=44,0.0.0.0 # set netbios-over-TCP/IP
nameserver(s) aka WINS server(s)

dhcp-option=45,0.0.0.0 # netbios datagram
distribution server

dhcp-option=46,8 # netbios node type
```

```
#dhcp-option=47

# VLAN DNS SERVER ID

dhcp-option=vlan10,6,8.8.8.8,8.8.4.4,192.168.10.1
# netbios node type

dhcp-option=vlan10,3,192.168.10.1

# TTL

#dhcp-option=23,64

# Broadcast Address

dhcp-option=28,192.168.10.255

# NTP Server

#dhcp-option=42,0.0.0.0

# Domain Name

dhcp-option=15,domain10.tld
```

### Configuration file for DMZ VLAN:

```
# Configuration file for dnsmasq OpenBsd Vlan Server.

interface=vlan20

listen-address=192.168.20.1

# Insert domain after hostname

domain=domain20.tld

expand-hosts

# Setting DHCP Ranges

dhcp-range=vlan20,192.168.20.20,192.168.20.200,255.255.25
5.0,48h

# Static DHCP Settings lan 10
```

```
#dhcp-host=00:0F:FE:33:CD:AE,192.168.20.3 # Asterisk
Server

#dhcp-host=00:48:54:5B:F8:EB,192.168.20.4 # VÅmware
Server

# SAMBA/WINBINDD Dhcp Server Options

dhcp-option=19,0          # option ip-forwarding off

dhcp-option=44,0.0.0.0    # set netbios-over-TCP/IP
nameserver(s) aka WINS server(s)

dhcp-option=45,0.0.0.0    # netbios datagram
distribution server

dhcp-option=46,8          # netbios node type

#dhcp-option=47

# VLAN DNS SERVER ID

dhcp-option=vlan20,6,8.8.8.8,8.8.4.4,192.168.20.1
# netbios node type

dhcp-option=vlan20,3,192.168.20.1

# TTL

#dhcp-option=23,64
```

```
# Broadcast Address

dhcp-option=28,192.168.20.255

# NTP Server

#dhcp-option=42,0.0.0.0

# Domain Name

dhcp-option=15,domain20.tld
```

### Configuration file for VOIP VLAN:

```
# Configuration file for dnsmasq OpenBsd Vlan Server.

interface=vlan30

listen-address=192.168.30.1
```



```
dhcp-option=15,domain30.tld
```

## VPN Concentrator:

A VPN (Virtual Private Network) is used when one needs to create a link between two or more private networks over a public network (like the Internet). Once the connection is established between the two private networks, users will see the opposite network in a completely transparent way as if they were physically connected to each other. But, you have to keep in mind that the maximum connection speed between the two networks is defined by the public network.

Another very important characteristic of the VPN is to create a secure communication system so you can rest assured even in case you need to transfer confidential data. All traffic between the two endpoints of the VPN is encapsulated in pre-tunnel. The tunnels can be on different levels of the ISO / OSI stack.

## pptp ed l2tp

```
# Insert domain after hostname

domain=domain30.tld

expand-hosts

# Setting DHCP Ranges

dhcp-range=vlan30,192.168.30.20,192.168.30.200,255.255.25
5.0,48h

# Static DHCP Settings lan 10

#dhcp-host=00:0F:FE:33:CD:AE,192.168.30.3 # Asterisk
Server

#dhcp-host=00:48:54:5B:F8:EB,192.168.30.4 # VÅmware
Server

# SAMBA/WINBINDD Dhcp Server Options

dhcp-option=19,0          # option ip-forwarding off

dhcp-option=44,0.0.0.0    # set netbios-over-TCP/IP
nameserver(s) aka WINS server(s)

dhcp-option=45,0.0.0.0    # netbios datagram
distribution server

dhcp-option=46,8          # netbios node type

#dhcp-option=47

# VLAN DNS SERVER ID

dhcp-option=vlan30,6,8.8.8.8,8.8.4.4,192.168.30.1
# netbios node type

dhcp-option=vlan30,3,192.168.30.1

# TTL

#dhcp-option=23,64

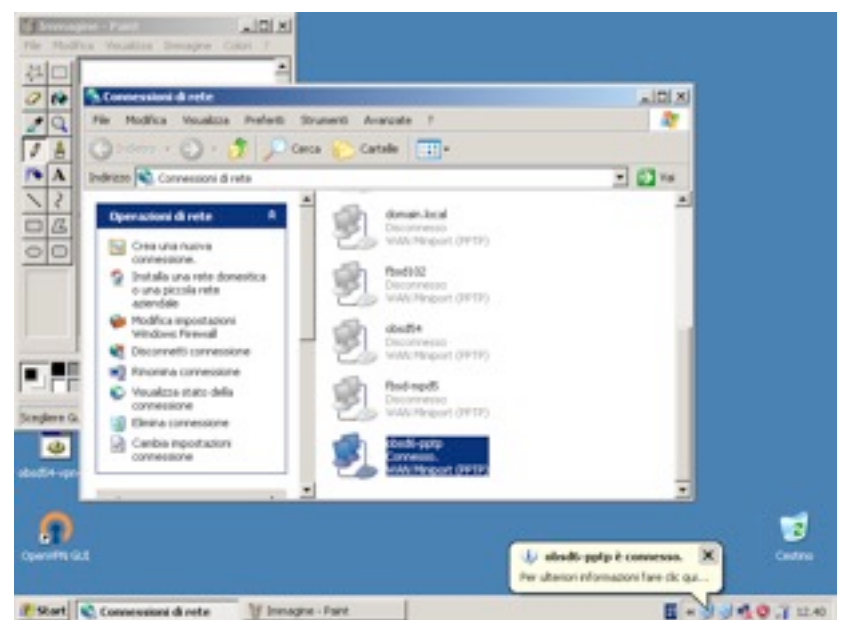
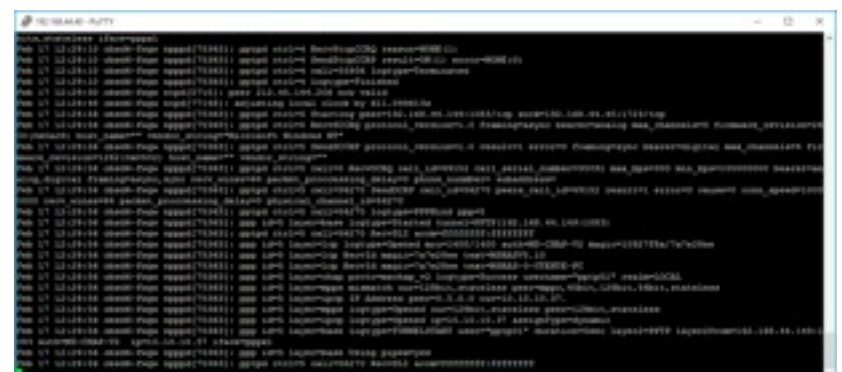
# Broadcast Address

dhcp-option=28,192.168.30.255

# NTP Server

#dhcp-option=42,0.0.0.0

# Domain Name
```



Main configuration file for L2TP and PPTP VPN server ( /etc/npppd/npppd.conf ):

```
authentication LOCAL type local {

    users-file "/etc/npppd/npppd-users"

}

tunnel L2TP protocol l2tp {

    listen on 0.0.0.0

#    mppe required

#    mppe-key-length 128

#    mppe-key-state stateless

#    listen on ::

}

ipcp IPCP-L2TP {

    pool-address 10.10.0.2-10.10.0.254

    dns-servers 8.8.8.8

}

tunnel PPTP protocol pptp {

    listen on 0.0.0.0

    pptp-vendor-name "openbsd-npppd"

    mppe required

    mppe-key-length 128

    mppe-key-state stateless

    idle-timeout 3600

}

ipcp IPCP-PPTP {

    pool-address 10.10.10.2-10.10.10.254

    dns-servers 8.8.8.8

}

interface pppx0 address 10.10.0.1 ipcp IPCP-L2TP

bind tunnel from L2TP authenticated by LOCAL to pppx0

interface pppx1 address 10.10.10.1 ipcp IPCP-PPTP

bind tunnel from PPTP authenticated by LOCAL to pppx1
```

Main configuration file for L2TP and PPTP VPN server user credentials ( /etc/npppd/ npppd-users):

```
#####

###

#

#                                PPTP Users PASSWORDS

#

#####

pptp01:\

                                :password=pptp01password:\

l2tp01:\

                                :password=l2tp01:\

#####

#

#                                L2TP Users PASSWORDS

#

#####
```

ipsec site to site ed ipsec/l2tp

Main configuration file for L2TP/IPSEC and SITE-TO-SITE VPN server ( /etc/ipsec.conf):

```
#####

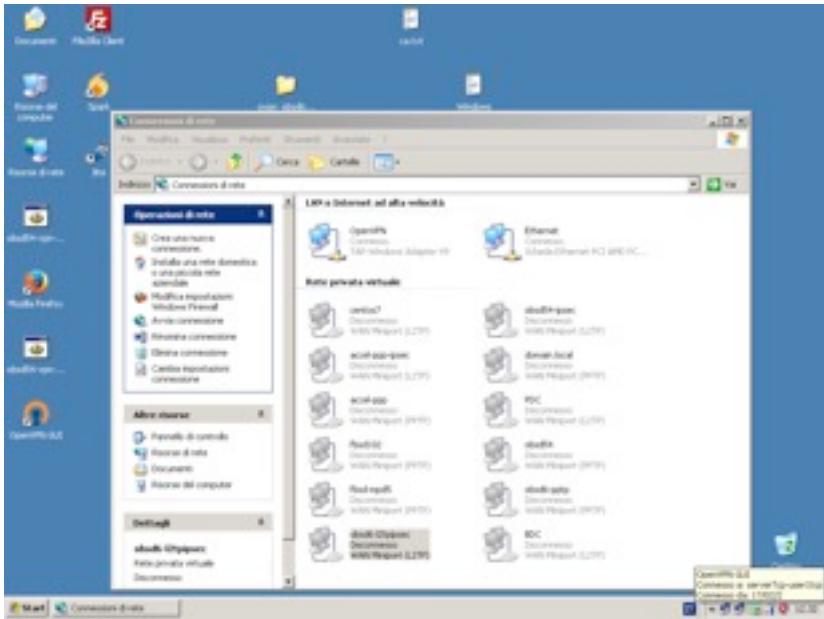
#

#

#                                IPSEC IKEv1 L2tp

#

#####
```





```

IF_WAN="em0"

key="ipsec12tpkey"

# Only for old OSes and Devices!!

ike passive esp transport \

    proto udp from $IF_WAN to any port 1701 \

    main auth hmac-sha1 enc 3des group modp1024 \

    quick auth hmac-sha1 enc 3des \

    psk $key

#####
###

#             IPSEC IKEv1 Site-to-Site
#

#####
###

local_ip="192.168.44.40"

local_network01="192.168.20.0/24"

local_network02="192.168.10.0/24"

remote_ip="192.168.44.50"

remote_network01="192.168.202.0/24"

remote_network02="192.168.203.0/24"

ike esp from $local_network01 to $remote_network01 peer
$remote_ip

ike esp from $local_network02 to $remote_network02 peer
$remote_ip

ike esp from $local_ip to $remote_network01 peer
$remote_ip

ike esp from $local_ip to $remote_network02 peer
$remote_ip

ike esp from $local_ip to $remote_ip

```

### OpenVPN Server for Road Warriors

```

useradd -d /home/openvpn/user1tcp -m -s /usr/bin/false
-g _openvpn -G _openvpnuers -p 'user1tcp123' user1tcp

```

**There are troubles as a result of using punctuation marks in password field.**

```

cd /etc/openvpn/udp/

```

```

cp -R /usr/local/share/easy-rsa/ .

cd easy-rsa/

cp vars.example vars

nano vars

./easyrsa init-pki

./easyrsa build-ca nopass

./easyrsa build-server-full server nopass

./easyrsa gen-dh

mkdir /etc/openvpn/udp/certskeys

cp /etc/openvpn/udp/easy-rsa/pki/private/server.key
/etc/openvpn/udp/certskeys/

cp /etc/openvpn/udp/easy-rsa/pki/issued/server.crt
/etc/openvpn/udp/certskeys/

cp /etc/openvpn/udp/easy-rsa/pki/ca.crt
/etc/openvpn/udp/certskeys/

cp /etc/openvpn/udp/easy-rsa/pki/dh.pem
/etc/openvpn/udp/certskeys/

```

### The /etc/openvpn/tcp/serverTcp.conf:

```

dev tun

proto tcp

port 443

## certs we created earlier

ca /etc/openvpn/tcp/certskeys/ca.crt

cert /etc/openvpn/tcp/certskeys/server.crt

key /etc/openvpn/tcp/certskeys/server.key

dh /etc/openvpn/tcp/certskeys/dh.pem

user _openvpn

group _openvpn

## You can make this any private subnet you like

server 10.192.0.0 255.255.255.0

persist-key

persist-tun

keepalive 10 120

comp-lzo

client-to-client

```

```

## make this connection the default gateway for network
traffic

#push "redirect-gateway def1"

#push "dhcp-option DNS 8.8.8.8"

push "route 192.168.10.0 255.255.255.0"

push "route 192.168.20.0 255.255.255.0"

push "route 192.168.30.0 255.255.255.0"

status /var/log/openvpn/openvpn-status.log

log-append /var/log/openvpn/udp.log

verb 3

client-cert-not-required

username-as-common-name

script-security 3 system

auth-user-pass-verify /usr/local/libexec/openvpn_bsdauth
via-env

#auth-user-pass-verify /usr/local/libexec/openvpn_bsdauth
via-file

## A management interface allows you to telnet from local
host to use

## telnet localhost 7505

#management localhost 7505

```

#### The /etc/openvpn/udp/serverUcp.conf:

```

dev tun

proto udp

port 1194

## certs we created earlier

ca /etc/openvpn/udp/certskeys/ca.crt

cert /etc/openvpn/udp/certskeys/server.crt

key /etc/openvpn/udp/certskeys/server.key

dh /etc/openvpn/udp/certskeys/dh.pem

user _openvpn

group _openvpn

## You can make this any private subnet you like

server 10.192.0.0 255.255.255.0

persist-key

persist-tun

```

```

keepalive 10 120

comp-lzo

client-to-client

## make this connection the default gateway for network
traffic

#push "redirect-gateway def1"

#push "dhcp-option DNS 8.8.8.8"

push "route 192.168.10.0 255.255.255.0"

push "route 192.168.20.0 255.255.255.0"

push "route 192.168.30.0 255.255.255.0"

status /var/log/openvpn/openvpn-status.log

log-append /var/log/openvpn/udp.log

verb 3

client-cert-not-required

username-as-common-name

script-security 3 system

auth-user-pass-verify /usr/local/libexec/openvpn_bsdauth
via-env

#auth-user-pass-verify /usr/local/libexec/openvpn_bsdauth
via-file

## A management interface allows you to telnet from local
host to use

## telnet localhost 7506

#management localhost 7506

```

#### Automated start of OpenVPN Server:

```

root@obsd6-fwgw[/etc]cat /etc/hostname.tun0

up

group openvpn

description "OpenVPN to local net 192.168.200.x"

!/usr/local/sbin/openvpn --daemon --config
/etc/openvpn/udp/serverUdp.conf --dev $if

root@obsd6-fwgw[/etc]cat /etc/hostname.tun1

up

group openvpn

description "OpenVPN to local net 192.168.200.x"

!/usr/local/sbin/openvpn --daemon --config
/etc/openvpn/tcp/serverTcp.conf --dev $if

```

## The client file clientTcp.ovpn:

```
### Sample client-side OpenVPN 2.0 config file.

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.

client

dev tun

proto tcp

# The hostname/IP and port of the server.

remote 192.168.44.40 443

# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.

resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.

nobind

comp-lzo adaptive

# Try to preserve some state across restarts.

persist-key

persist-tun

# Certificate Authority

;ca ca.crt

# Username/Password authentication is used on the server

;auth-user-pass

;ns-cert-type server

auth-user-pass login.txt

# Set log file verbosity.

verb 3

; For Windows 7 and 8 systems

route-method exe

route-delay 2

<ca>
```

```
-----BEGIN CERTIFICATE-----
```

```
MIIDJDCCAgYgAwIBAgIJAIkhwzEGfePJMA0GCSqGSIb3DQEBCwUAMBExD
zANBgNV

BAMMBnNlcnZlcjAeFw0xNjEyMjExMDI5MTJaFw0yNjEyMTkxMDI5MTJaM
BExDzAN

BgNVBAMMBnNlcnZlcjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCg
gEBANEp

9xJ9Ru3JcopA5qU39vxPHZqwM2mdLvA5b8tA+An9IYqk/wvM/Aviv/Wb4
c9Zl5dX

mH3tcNxELnlqbbhiwwbTzQi9HlcvPPhAJhXAlOi92awhKMw43VdKioic4
vsc2fiR

+czuD/h17BQcgn5j2WpUvBBzbKF6APzkX+j6GNVxsWppZ0QHL9aCQ3Fdh
MKzHVD4

sc0JSaqq+f9z50qeZ/ZG5RF/e55bfoyW+smAts+eMsaoJLcBRbxnSCcEv
uzwomIC

WoVM6ULVsedXxXqggjjaJ8QTG0ynmGr+B8deVWYgXivAllAOjh2zYi5Ur
87dT8ni

zskRLAgwSqTPk+CwXI0CAwEAAaN/MH0wHQYDVR0OBBYEFGWCO0pAOpJiQ
z7/hjm9

T23qp/DwMEEGA1UdIwQ6MDiAFGWCO0pAOpJiQz7/hjm9T23qp/DwoRWkE
zARMQ8w

DQYDVQQDDAZzZXJ2ZXKCCQCIcMxBn3jyTAMBgNVHRMEBTADAQH/MASGA
1UdDwQE

AwIBBjANBgkqhkiG9w0BAQsFAAOCAQEAAUxNkoLHUnT5GFVZqW4nJOcZv/
jWMK9QS

Sk+6WjuMhyYG6esz9X3WF0M8K8tXlmOL6w6NLn9cj6APs7C8liWdccbR8
UylIPJo

amcax/Cmpr5lZwYLgPScwBEi6SMScb/lxdPEKdqCQbNeZwIkQ1lBfKziN
nhXu/5n

PSWAKWOOYQRQnosU8HC7IDRLwnLAgl7oNdrwM99M3TT3534T0kIX+4SqR
zpZbbQf

kqlaGB01zT9SzsOYB5gkwLMYdZcgVGMaiHFiya0elW7/lx7oN9AEEZqX8
nM+pCVL

HlJHfjRGShC6miZXTgNMPX5B7GBrzDV8EgacTqdmowPNCkPCKkXePw==

-----END CERTIFICATE-----

</ca>
```

## The client password file password.txt:

```
user1tcp

user1tcp123
```



## Firewall complete script

After you have installed and configured all the services seen so far, it remains to complete the final step, the configuration of the firewall Packet Filter, which will act as "glue" and arbiter for all the connections of the company network.

```
#####

#      pf.conf openbsd 6.0 domain.tld      #
#      with vlan/nat/dhcp/dns services      #
#####

#####

# macros section start  #

#####

ext_if="em0"

ext_ip="192.168.44.40"

ext_net="192.168.44.0/24"

int_if="vlan10"

dmz_if="vlan20"

voip_if="vlan30"

lan_lan="192.168.10.0/24"

dmz_lan="192.168.20.0/24"

voip_lan="192.168.30.0/24"

tun_ifs="{ tun0, tun1, tun2, tun3, tun4, tun5, tun6,
tun7, tun8, tun9 }"

pppx_ifs="{ pppx0, pppx1, pppx2, pppx3, pppx4, pppx5,
pppx6, pppx7, pppx8, pppx9 }"

tcp_services="{ 22, 113, 53 }"

udp_services="{ 53, 113 }"

icmp_types="echoreq"

web_services = "{80, 21, 443}"

google_ns="{8.8.8.8, 8.8.4.4}"

obsd_repo="193.206.140.37" #openbsd.mirror.garr.it

pptp_services = "{ 1723, 47 }"

ipsec_services = "{ 50, 500, 4500 }"
```

```
ovpn_services = "{ 1194 }"

l2tp_services = "{ 1701, 17 }"

### Tables

table <blocked> persist file "/etc/blocklist"

table <bruteforce> persist

# IPSEC VPN TABLES

table <vpn_peers> const { 192.168.44.50 }

#####

### Matching Internal/external IPs for Natting ...

#####

# macros section end      #

#####

#####

# options section start #

#####

set block-policy return

set loginterface $ext_if

set skip on { lo, enc0, gre0 }

# FTP Proxy rules

anchor "ftp-proxy/*"

pass on $int_if inet proto tcp from $ext_if port > 1023
to port ftp divert-to 127.0.0.1 port 8021

pass on $dmz_if inet proto tcp from $ext_if port > 1023
to port ftp divert-to 127.0.0.1 port 8021

pass on $ext_if inet proto tcp from $ext_if port > 1023
to port ftp divert-to 127.0.0.1 port 8021
```

```
# ftp out from ftp-proxy and local machine, allow passive
ftp out
```

```
pass out log on $ext_if inet proto tcp from ($ext_if)
port > 1023 to any port ftp modulate state
```

```
pass out log on $ext_if inet proto tcp from ($ext_if)
port 54999><56999 to any port > 1023 modulate state
```

```
pass out log on $ext_if inet proto tcp from ($ext_if)
port > 1023 to any port > 1023 modulate state
```

```
pass in log on $dmz_if inet proto tcp from $ext_if port >
1023 to any port > 1023 modulate state
```

```
pass in log on $int_if inet proto tcp from $ext_if port >
1023 to any port > 1023 modulate state
```

```
pass in log on $ext_if inet proto tcp from $ext_if port >
1023 to any port > 1023 modulate state
```

```
#####
```

```
# options section end  #
```

```
#####
```

```
# <-- NAT SECTION START  --> #
```

```
#####
```

```
# nat rules start  #
```

```
#####
```

```
### DMZ NAT 1:1
```

```
## Host : fw.domain10.local
```

```
fw_srv_int="192.168.20.1"
```

```
fw_srv_ext="192.168.44.40"
```

```
pass in on $ext_if inet proto tcp to $fw_srv_ext port
{53,22,953,443,1723} rdr-to $fw_srv_int keep state
```

```
pass in on $ext_if inet proto udp to $fw_srv_ext port
{53,953} rdr-to $fw_srv_int keep state
```

```
pass out quick log on $ext_if inet from $fw_srv_int to
any nat-to $fw_srv_ext
```

```
## Host : www1.domain20.local
```

```
www1_srv_int="192.168.20.40"
```

```
www1_srv_ext="192.168.44.41"
```

```
pass out quick log on $ext_if inet from $www1_srv_int to
any nat-to $www1_srv_ext
```

```
pass in log quick on $ext_if inet proto tcp to
$www1_srv_ext port 80 rdr-to $www1_srv_int port 8000
```

```
# rdr-to firewall IP
```

```
#pass in log on $ext_if inet proto tcp to ($ext_if) port
80 rdr-to $www1_srv_int port 8000
```

```
pass in log quick on $ext_if inet proto tcp to
$www1_srv_ext port {22,443} rdr-to $www1_srv_int keep
state
```

```
### Host : asterisk.domain20.local
```

```
asterisk_srv_int="192.168.20.4"
```

```
asterisk_srv_ext="192.168.44.42"
```

```
pass in on $ext_if inet proto tcp to $asterisk_srv_ext
port {22,443,5222,7777,9090,9091,5060} rdr-to
$asterisk_srv_int keep state
```

```
pass in on $ext_if inet proto udp to $asterisk_srv_ext
port
{5060,4596,5036,2727,5222,7777,9090,9091,9999}<20001}
rdr-to $asterisk_srv_int keep state
```

```
pass out quick log on $ext_if inet from $asterisk_srv_int
to any nat-to $asterisk_srv_ext
```

```
# binat-to example
```

```
#pass quick log on $ext_if from $presenze_srv_int to any
binat-to $presenze_srv_ext
```

```
#####
```

```
# match rules start  #
```

```
#####
```

```
### LANs NAT MANY:1
```

```
match out on $ext_if from $lan_lan nat-to ($ext_if)
```

```

match out on $ext_if from $dmz_lan nat-to ($ext_if)

match out on $ext_if from $voip_lan nat-to ($ext_if)

#####

#   match rules end       #

#####

# <-- NAT SECTION END  --> #

#####

#   filter rules start   #

#####

block in log

block log all

#####

#   Global Block Section start

#####

#block in quick on $ext_if inet proto tcp from any to
$nasl3_srv_int port 21

#####

#   Global Block Section End

#####

pass out quick

pass quick on $ext_if from $ext_net

antispoof quick for { lo $int_if $dmz_if $voip_if}

block in log on $ext_if from <blocked> to any

block quick from <bruteforce>

pass in on $ext_if inet proto tcp from any to $ext_ip
port $tcp_services

pass in on $ext_if inet proto udp from any to $ext_ip
port $udp_services

pass out on $ext_if inet proto tcp from $ext_ip to
$obsd_repo port {80,21} keep state

pass in on $ext_if inet proto tcp from $obsd_repo to
$ext_ip port {80,21} keep state

## enable Google public NS

pass in on $ext_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass out on $ext_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass in on $int_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass out on $int_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass in on $dmz_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass out on $dmz_if proto {tcp, udp} from $google_ns to
any port 53 keep state

pass in on $ext_if proto tcp from any to any port 21 keep
state

pass in on $ext_if proto tcp from any to any port > 49151
keep state

pass out on $ext_if proto tcp from any to any port 21
keep state

pass out on $ext_if proto tcp from any to any port >
49151 keep state

pass in on $ext_if proto tcp from $obsd_repo to any port
{80,21} keep state

pass out on $ext_if proto tcp from $obsd_repo to any port
{80,21} keep state

pass in on $int_if proto tcp from $obsd_repo to any port
{80,21} keep state

pass out on $int_if proto tcp from $obsd_repo to any port
{80,21} keep state

pass log inet proto tcp from any to any port ssh flags
S/SA keep state (max-src-conn 5, max-src-conn-rate 5/30,
overload <bruteforce> flush global)

```



```

#####

#   filter rules end   #

#####

#####

#   final ruleset start   #

#####

##### Vpn Section Start

##### OpenVpn Ruleset

pass in on $ext_if proto udp from any to any port 1194

pass out on $ext_if proto udp from any to any port 1194

pass in quick proto udp from any to port 1194 keep state
label "Openvpn"

pass in quick on $ext_if proto udp from any to $ext_ip
port = 1194 modulate state

pass in on $tun_ifs inet proto {tcp,udp,icmp} from any to
any flags S/SA keep state

pass out on $tun_ifs inet proto {tcp,udp,icmp} from any
to any flags S/SA keep state

pass in quick log on $tun_ifs keep state

pass out quick log on $tun_ifs keep state

##### Pptpd/L2tpd Ruleset

#Pptp Vpn Rules

pass in quick on $ext_if proto tcp from any to $ext_ip
port = 1723 modulate state label "Pptpd"

pass in quick on $ext_if proto gre from any to $ext_ip
keep state

pass out quick on $ext_if proto gre from $ext_ip to any
keep state

pass in quick log on $pppx_ifs all

pass out quick log on $pppx_ifs all

#L2tp Vpn Rules

pass in quick on $ext_if proto udp from any to any port
{17, 1701} keep state label "L2tpd"

pass on enc0 from any to any keep state (if-bound)

pass in quick log on $tun_ifs all

pass out quick log on $tun_ifs all

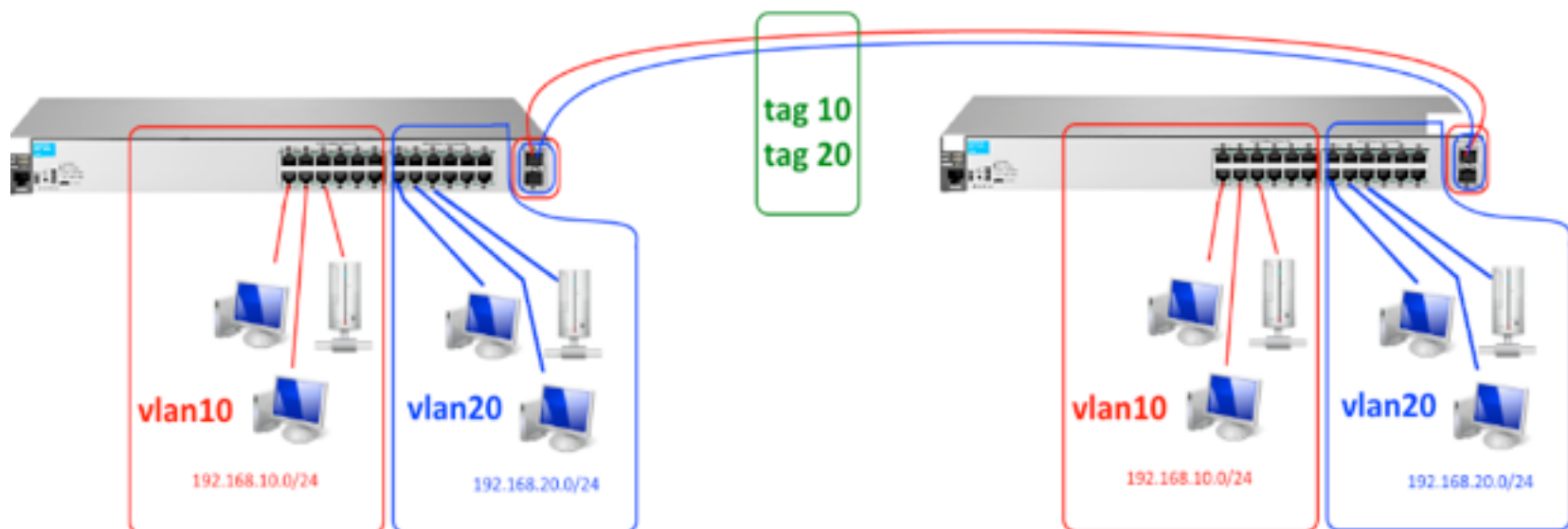
##### Ipsec/L2tpd Ruleset

pass quick proto { esp, ah } from any to any

pass in quick on $ext_if proto udp from any to any port
{500, 4500, 1701} keep state label "IpsecL2tpd"

pass on enc0 from any to any keep state (if-bound)

```



```
##### Ipsec Site-to-Site Ruleset

pass out quick on egress proto esp from (egress:0) to
<vpn_peers> keep state

pass out quick on egress proto udp from (egress:0) to
<vpn_peers> port {500, 4500} keep state

pass in quick on egress proto esp from <vpn_peers> to
(egress:0) keep state

pass in quick on egress proto udp from <vpn_peers> to
(egress:0) port {500, 4500} keep state

##### Vpn Section End

pass in inet proto icmp all icmp-type $icmp_types

pass in on $int_if

pass in on $dmz_if

#####

# final ruleset end #

#####
```

### Cisco Switch simple configuration for Main office Vlans:

```
enable

vlan database

vlan 10 name MainOffice-VlanLan

vlan 20 name MainOffice-VlanDMZ

vlan 30 name MainOffice-VlanVoip

exit

conf t

hostname MainSwitch

interface FastEthernet 1/0

switchport trunk encapsulation dot1q

switchport mode trunk

switchport trunk allowed vlan all

no shutdown

interface range fastethernet1/1 - 10

switchport mode access
```

```
switchport access vlan 30

no shutdown

interface range fastethernet1/11 - 15

switchport mode access

switchport access vlan 20

no shutdown

interface range fastethernet2/1 - 15

switchport mode access

switchport access vlan 10

no shutdown

exit

copy run start
```

## Conclusions

As we have seen, OpenBSD makes possible the creation of a Gateway Router Firewall and a multi VPN concentrator. The level of security it can give is very high both for SOHO or enterprise infrastructures. Therefore, OpenBSD proves to be a great alternative to using a dedicated and expensive hardware equipment, plus it's open-source.

## About the Author

Antonio Francesco Gentile is a software and network engineer who lives in Italy, Calabria. He works for CNR, the National research center as a Network Manager, with the "Culture Lab" <http://culture.deis.unical.it> Department of Telematics at University of Calabria, the computer science associations "Hacklab Cosenza" <https://hlcs.it/> and "Verde Binario" <http://www.verdebinario.org/>. Additionally, he is a freelance columnist for Italian magazines such as

"Linux & C" <http://www.oltrelinux.com/>,  
 "LinuxMagazine" <http://www.linux-magazine.it/> and  
 "Elettronica OpenSource2" <http://it.emcelettronica.com/>.

# FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.



## HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

## NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



*Example of one-bit corruption*

## THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

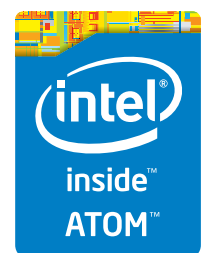
No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

### The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.iXsystems.com/mini>





# FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

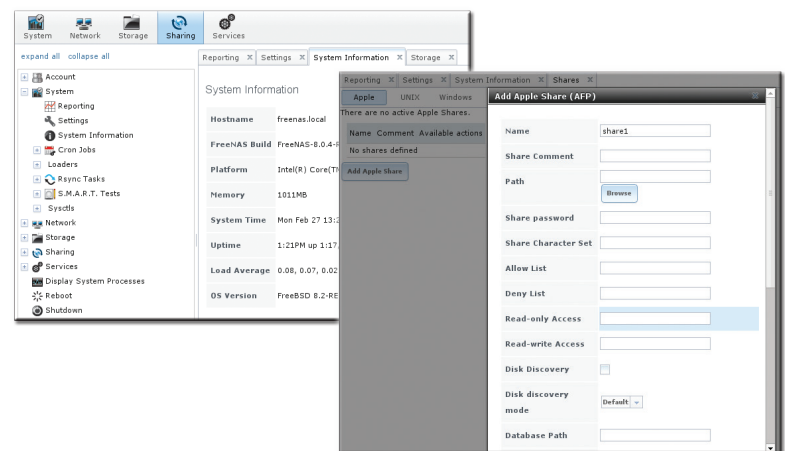
## MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

## Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



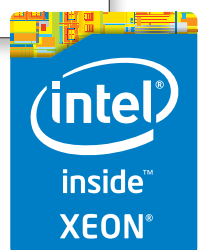
### FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

### FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

<http://www.iXsystems.com/storage/freenas-certified-storage/>



# OPNsense

OPNsense is an open-source, easy-to-use and easy-to-build FreeBSD based firewall and routing platform. OPNsense includes most of the features available in expensive commercial firewalls, and more in many cases. It brings a rich feature set of commercial offerings with the benefits of open and verifiable sources.

OPNsense started as a fork of pfSense® and m0n0wall in 2014, with its first official release in January 2015. The project has evolved faster while still retaining familiar aspects of both m0n0wall and pfSense. A strong focus on security and code quality drives the development of the project. Weekly, OPNsense offers security updates with small adjustments to react on new emerging threats within in a fashionable time. A fixed release cycle, two major releases each year, offers businesses the opportunity to plan upgrades ahead. For each major release, a roadmap is installed to guide the development and in setting out clear goals. List of OPNsense features:

- Traffic Shaper
- Two-factor authentication throughout the system.
- A captive portal.
- Forward Caching Proxy (transparent) with Blacklist support.
- Virtual Private Network (site-to-site and road warrior, IPsec, OpenVPN & legacy PPTP support).
- High availability and Hardware Failover ( with configuration, synchronization and synchronized state tables)
- Intrusion Detection and Prevention mechanism.
- Built-in reporting and monitoring tools including RRD Graphs.

The main differences between an embedded image and a full image are:

Embedded	Full
Uses NanoBSD	Uses FreeBSD
Writes to RAM disk	Writes to local disk
No log data retention after reboot	Log data retention after reboot
Not intended for local disk writes	Suitable for disk writes.
Embedded only use	Can enable RAM disk for embedded mode.

- Netflow Exporter.
- Network Flow Monitoring.
- Support for plug-ins.
- DNS Server & DNS Forwarder.
- DHCP Server and Relay.
- Dynamic DNS.
- Encrypted configuration backup to Google Drive.
- Stateful inspection firewall.
- Granular control over state table.
- 802.1Q VLAN support.

The feature set of OPNsense includes high-end features such as forward caching proxy, traffic shaping, intrusion detection and easy OpenVPN client setup. The latest release is based upon FreeBSD 10.2 for long-term support, and uses a newly developed MVC-framework based on Phalcon.

OPNsense’s focus on security brings unique features such as the option to use LibreSSL instead of OpenSSL (selectable in the GUI), and a custom version based on HardenedBSD.

The robust and reliable update mechanism gives OPNsense the ability to provide important security updates in a timely fashion.

The software setup and installation of OPNsense® is available for x86-32 and x86-64 bit microprocessor architectures.

### Embedded vs. Full

Full installs can run on SD memory cards, solid-state disks (SSD) or hard disk drives (HDD).

Since the release of version 15.1.10 (04 May 2015), the option to install an embedded OPNsense image is also supported. Embedded images (NanoBDS) store logging and cache data in memory only, while Full versions keep the data stored on the local drive. A full version can mimic the behaviour of an Embedded version by enabling RAM disks, this is especially useful for SD memory card installations.

## OPNsense VS Pfsense

They have numerous similarities. When you setup OPNsense for first time, you might think it's Pfsense with a different GUI. In fact, they are similar. However, in my opinion, Pfsense GUI is easier to use but OPNsense has a faster GUI.

Advantages of OPNsense:

- Faster GUI(lighttpd).
- Better Security Update.
- UEFI Support (OPNsense supports UEFI and Pfsense "not").

More Clear License (OPNsense is available under the BSD 2-Clause "Simplified" or "FreeBSD" license:)

## Virtual Firewall

A **virtual firewall (VF)** is a network firewall service or appliance running entirely within a virtualized environment and offers the usual packet filtering, and monitoring provided via a physical network firewall. Virtualized firewall is not the best solution. A separated firewall appliance is more convenient but setup VF is more cost effective. In this section, we will setup OPNsense on Bhyve. Bhyve (pronounced "bee hive", formerly written as BHyVe) is a type-2 hypervisor/virtual machine manager for FreeBSD. It was introduced in FreeBSD 10.0 and supports most Intel and AMD processors that report the "POPCNT" (POPulation Count) processor feature in dmesg(8). The Bhyve BSD-licensed hypervisor became part of the base system with FreeBSD 10.0-RELEASE. This hypervisor supports a number of guests, including FreeBSD, OpenBSD and many Linux distributions. Currently, Bhyve only supports a serial console and does not emulate a graphical console. Virtualization offload features of newer CPUs are used to avoid the legacy methods of translating instructions and manually managing memory mappings. The Bhyve design requires a processor that supports

Intel Extended Page Tables (EPT), AMD Rapid Virtualization Indexing (RVI) or Nested Page Tables (NPT). It runs FreeBSD 9+, OpenBSD, NetBSD, Linux and MS Windows desktop (versions Vista, 7, 8/8.1/8.2 and 10), as well as MS Windows Server (versions 2008/2008R2, 2012/2012R2 and 2016 Technical Preview 2 and 3) guests.

Lately, libvirt supports Bhyve as well. But personally, I prefer to utilize Bhyve from the shell. Also, there are FreeBSD packages that were created to make life easier like CBSD and VM-Bhyve.

Recently, the Bhyve hypervisor supports Unified Extensible Firmware Interface Graphics Output Protocol or "UEFI-GOP". It means that you can easily run any modern OS without pain.

## OPNsense installation on Bhyve

Presently, Bhyve supports UEFI-GOP in FreeBSD 11.0-RELEASE.

OPNsense requirements:

- Minimum required RAM is 1 GB.
- Minimum recommended virtual disk size of 8GB.

1. Install FreeBSD 11.0. You can also install FreeBSD 11.0 or any latest builds.

2. Retrieve the firmware binary. We must first install "bhyve-firmware". The best way to achieve this goal is to install with port mechanism. This process is very time-consuming and demands intense user-interaction. Nonetheless, with some tricks, it can be easily done with the following commands:

```
# cd /usr/ports/sysutils/bhyve-firmware
```

```
# make install clean -DBATCH
```

-DBATCH forces a port building process to not prompt you for confirmation and does it automatically.

3. Hypervisor, Network and Storage Preparation

```
# kldload vmm
```

This command will either load bhyve kernel module or a driver.

```
# ifconfig tap0 create up
```



This command creates a new interface and makes it up.

```
# ifconfig bridge0 create up
```

This command also creates a bridge, makes it up and ready to use.

```
# ifconfig bridge0 addm em0
```

This command adds em0(network interface) to bridge0

```
# ifconfig bridge0 addm tap0
```

This command adds tap0 to bridge0.

```
# truncate -s 50G OPNsense.img
```

This command creates a file with 50GB size.

#### 4. Prepare OPNsense ISO

```
#fetch
```

```
http://mirror.ams1.nl.leaseweb.net/opnsense/releases/mirror/OPNsense-17.1-OpenSSL-cdrom-amd64.iso.bz2
```

```
# bunzip2
```

```
OPNsense-17.1-OpenSSL-cdrom-amd64.iso.bz2
```

#### 5. Boot a Virtual Machine

```
# bhyve -c 2 -m 4G -w -H \
```

```
-s 0,hostbridge \
```

```
-s
```

```
3,ahci-cd,OPNsense-17.1-OpenSSL-cdrom-amd64.iso \
```

```
-s 4,ahci-hd,OPNsense.img \
```

```
-s 5,virtio-net,tap0 \
```

```
-s 29,fbuf,tcp=0.0.0.0:5900,w=800,h=600,wait \
```

```
-s 30,xhci,tablet \
```

```
-s 31,lpc -l com1,stdio \
```

```
-l
```

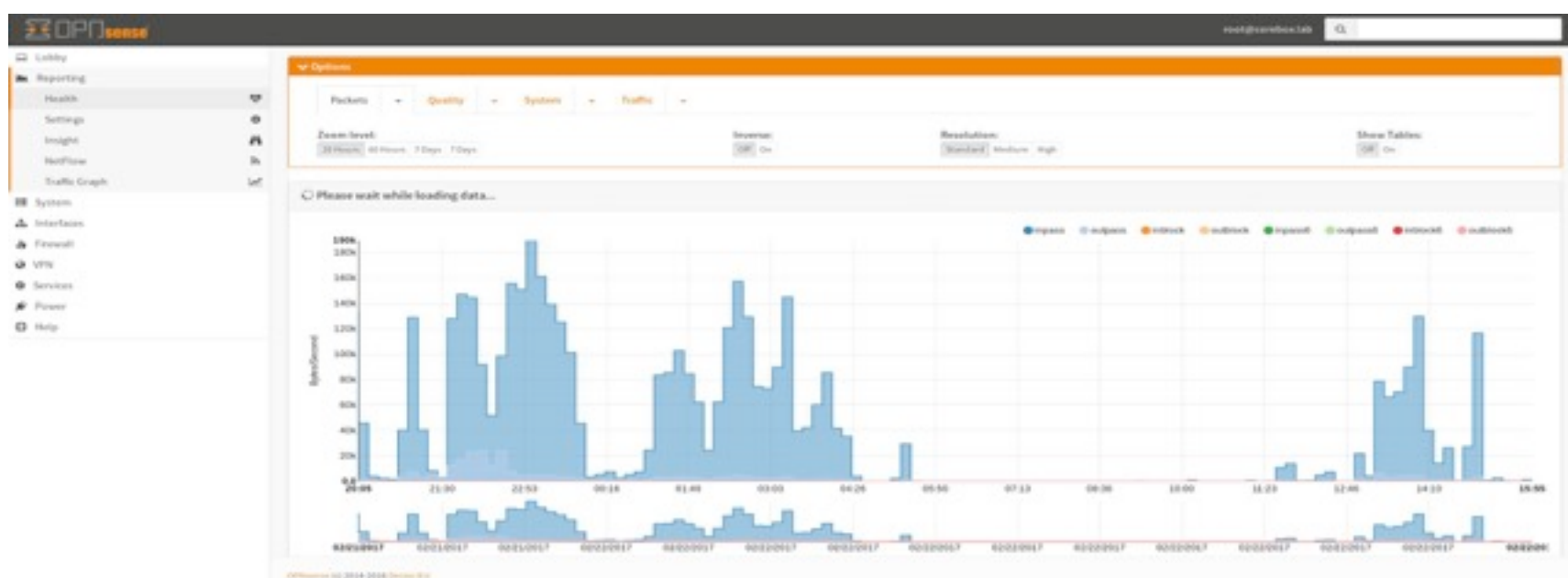
```
bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
```

#### OPNsense

This command makes a virtual machine(vm0) with two cores CPU, and with a display resolution of 800 by 600 that can be accessed via VNC at: 0.0.0.0:5900. The fbuf wait parameter instructs Bhyve to only boot upon the initiation of a VNC connection, simplifying the installation of operating systems that require immediate keyboard input. This can be removed for post-installation use. The xhci,tablet parameter provides precise cursor synchronization when using VNC, but it is not supported by FreeBSD.

**-H** Yields the virtual CPU thread when a HLT instruction is detected. If this option is not specified, virtual CPUs will use 100% of a host CPU.

**-w** Ignores accesses to unimplemented Model Specific Registers (MSRs). This is intended for debugging purposes.



6. Connect to VM with VNC client

# vncviewer 192.168.1.1:5900

In VNC Client screen, you can see what happening. Also, mouse are supported. I prefer to use “tightvnc” with my hypervisor IP as “192.168.1.1”.

7. Installation process:

- Configure console - The default configuration should be fine for most occasions.
- Select task - The **Quick/Easy Install** option should be fine for most occasions. For installations on embedded systems or systems with minimal disk space, choose **Custom Installation** and do not create a swap slice. Proceed with default settings.
- **Are you SURE?** - When proceeding OPNsense, will be installed on the **first hard disk** in the system.
- Reboot - The system is now installed and needs to be rebooted to continue with configuration.

### Initial configuration

After installation the system will prompt you for the interface assignment. If you ignore this, default settings are applied. Installation ends with the login prompt. By default, you have to log in to enter the console.

### Welcome message

```
* * * Welcome to OPNsense [OPNsense 15.7.25
(amd64/OpenSSL) on OPNsense * * *
```

```
WAN (em1)  ->
```

```
LAN (em0)  -> v4: 192.168.1.1/24
```

```
FreeBSD/10.1 (OPNsense.localdomain) (ttyv0)
```

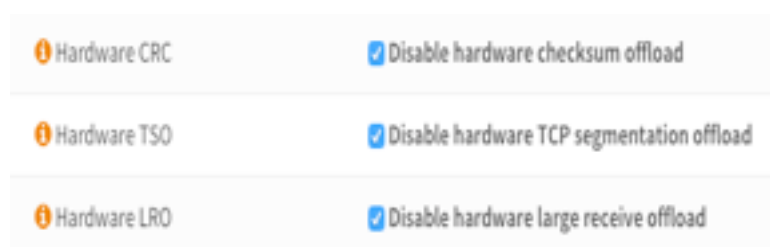
```
login:
```

### OPNsense Mandatory Configuration

A user can login to the console menu with his credentials. The default credentials after a fresh install are username “root” and password “opnsense”.



For better performance under virtual environment, disable all off-loading settings in **System->Settings->Networking** As you can see, OPNsense has very beautiful GUI that lets you control every single aspect of the firewall.

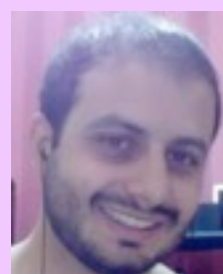


### Conclusion

OPNsense is very easy to use on your virtual infrastructure. With UEFI support, you can install OPNsense on a modern mainboard.

### Useful Links

<https://docs.opnsense.org/manual/virtuals.html>  
<http://in4bsd.com/page/FreeBSD>



### About the Author

Abdorrahman Homaei has been working as a software developer since 2000. He used FreeBSD for more than ten years. He became involved with the meetBSD dot ir and performed serious trainings on FreeBSD. He is starting his own company in February 2017. You can visit his site to view his curriculum vitae: <http://in4bsd.com>

# Kill a Long Running Process in Unix

**How To - Kill a long running process in Unix.**

**<http://www.ivishal.me>**



**Vishal Lambe is a tech blogger, author, part-time cartoonist, caricaturist and a voracious reader. He practices software engineering for a living. Moreover, he is a PMI Certified Associate in Project Management and has authored two books.**

A UNIX process runs in the background and keeps on performing the allocated task. However, when the process runs for a long time, wait for some other resource or get stuck in deadlock, it continuously consumes memory. At such times, we may want to release the memory and other resources by killing such processes.

To kill a particular process, we need to determine its PID, ie. Process ID. This can be done using the 'ps' command. A 'ps' command takes the snapshot of the memory when it runs and lists all the running process statistics. This is the very reason for including itself in the list.

```
$ ps
```

The next step is to use the 'kill' command along with the PID of the process you wish to kill. You can specify the 'kill' command to be non-maskable. Using -9 with the 'kill' command makes it non-maskable.

```
$ kill -9
```

Suppose you have a long running process - 'grep' in Unix which you desire to purge or kill. You'll first need to find out the PID of this process.

```
dev@pb456-5878:/app/data/vishal> ps
```

```
PID TTY    TIME CMD
```

```
1983 pts/44  00:00:00 more
```

```
2635 pts/44  00:00:00 vi
```

```
6521 pts/44 00:00:00 more
9622 pts/44 00:00:00 ps
12458 pts/44 00:00:00 more
12605 pts/44 00:00:00 man
12618 pts/44 00:00:00 sh
12627 pts/44 00:00:00 less
13084 pts/44 00:00:00 cat
13161 pts/44 00:00:00 more
15732 pts/44 00:00:00 more
15933 pts/44 00:00:00 more
16603 pts/44 00:00:00 more
17444 pts/44 00:00:00 grep
17930 pts/44 00:00:00 more
19905 pts/44 00:00:00 more
20264 pts/44 00:00:00 more
21976 pts/44 00:00:00 more
27540 pts/44 00:00:01 bash
31061 pts/44 00:00:00 ksh
32137 pts/44 00:00:00 more
```

Now, use the below command to kill the 'grep' process.

```
dev@pb456-5878:/app/data/vishal> kill -9 17444
```

```
[7] Killed grep garantia (wd:
dev@pb456-5878:/app/data/vishal)
```

```
(wd now: dev@pb456-5878:/app/data/vishal)
```

You can verify the successful purge of the 'grep' process by issuing the 'ps' command again.

```
dev@pb456-5878:/app/data/vishal> ps
```

```
PID TTY    TIME CMD
```

```
1983 pts/44 00:00:00 more
2635 pts/44 00:00:00 vi
6521 pts/44 00:00:00 more
9622 pts/44 00:00:00 ps
12458 pts/44 00:00:00 more
12605 pts/44 00:00:00 man
12618 pts/44 00:00:00 sh
12627 pts/44 00:00:00 less
13084 pts/44 00:00:00 cat
13161 pts/44 00:00:00 more
15732 pts/44 00:00:00 more
15933 pts/44 00:00:00 more
16603 pts/44 00:00:00 more
17930 pts/44 00:00:00 more
19905 pts/44 00:00:00 more
20264 pts/44 00:00:00 more
21976 pts/44 00:00:00 more
27540 pts/44 00:00:01 bash
31061 pts/44 00:00:00 ksh
32137 pts/44 00:00:00 more
```

You can see in the above list that 'grep' is not present.



## **Interview with Vishal Lambe**

### **Can you tell our readers about yourself and your blog?**

I am a data-warehouse professional and a part-time blogger. I work with Unix, Informatica, SQL, Hive and Git.

My favorite personality is RMS. I am the author of book - *The Gita Of Programming - Lord Krishna's Teachings on the ethics of Programming*.

I blog @ [www.ivishal.me](http://www.ivishal.me)

### **How you first got involved in blogging?**

I started blogging during my college days. Thereafter, I got more inclined towards writing on Unix. In general, Unix has numerous applications in ETL, databases and Data warehousing.. This is where my job becomes a sub-set of my hobby!

### **What's the best thing a blogger can give to his readers?**

Giving what the readers want. Understanding reader's appetite and publishing timely articles is the key to a successful blog.

### **Everyone has a favorite/least favorite post. Name yours and why?**

My popular Unix post is "less vs more vs vi". All posts are close to my heart, I really cannot think of a least favorite one.

<http://www.ivishal.me/2014/08/unix-terminal-hanged-her-e-what-you-need.html>

### **What do you think what makes Unix so beloved by programmers?**

Because of its simplicity. "UNIX is user-friendly; it just chooses its friends." - Andreas Bogk

Like a good Bob Dylan song, Unix grows on you over a period. You'll get attached to it.

Although it may at first seem troublesome, most people have found that command line operation becomes quite easy, and even somewhat intuitive, with practice. Unix CLI is the most important feature of Unix and loved by most of the programmers.

### **What is your advice to anyone who wants to advance their UNIX knowledge?**

First, know your basics. Do not jump to internal Unix complexities directly. Master the command line and you will save a lot of time.

Unix takes its own sweet time to settle. Assuming that you are sufficiently motivated and have good study habits, plenty of excellent resources are available online as well as offline for learning Unix. You can learn at your own pace. Becoming a true Linux guru can take years of study and experience.

### **What is your favorite OS and why?**

For personal computing, I prefer FreeBSD. Before being introduced to BSD flavors, I liked Linux Mint. Your personal favorite depends a lot on your taste and usage.

### **What is the future of UNIX in general or your favourite OS? What do you think?**

Unix has left no industry untouched. I work in Finance domain and my personal experience over the years is that no application is complete without the elegance of Unix. Be it legacy applications or a complex interface between multiple applications. Unix is an architect's first choice for reliability and sustainability due its backward compatibility.

### **Do you have any specific goals for the rest of this year?**

My 2017 goal is to build interesting applications in Spark and Python.

# Among clouds Performance and Reliability is critical

Download syslog-ng Premium Edition  
product evaluation [here](#)

Attend to a free logging tech webinar [here](#)



**BalaBit**  
IT Security

[www.balabit.com](http://www.balabit.com)

## syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

### HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

# Shell From vi

vi is a good example of a tool that interacts openly and easily with the Unix shell  
<https://sanctum.geek.nz>

**Tom Ryder is a systems administrator and programmer living in New Zealand. On his blog he posts articles on systems administration and programming, particularly where it relates to his interests in Unix, GNU/Linux, shell scripting, C, Perl, Vim, Git, or whatever else takes his interest from a technical bent. A favorite topic is using command-line tools effectively and efficiently.**

A good sign of a philosophically sound interactive Unix tool is the facilities it offers for interacting with the filesystem and the shell: specifically, how easily can you run file operations and/or shell commands with reference to data within the tool? The more straightforward this is, the more likely the tool will fit neatly into a terminal-driven Unix workflow.

If all else fails, you could always suspend the task with Ctrl+Z to drop to a shell, but it's helpful if the tool shows more deference to the shell than that; it means you can use and (even more importantly) *write* tools to manipulate the data in the program in whatever languages you choose, rather than being forced to use any kind of heretical internal scripting language, or worse, an over-engineered API.

vi is a good example of a tool that interacts openly and easily with the Unix shell, allowing you to pass open buffers as streams of text transparently to classic filter and text processing tools. In the case of Vim, it's particularly useful to get to know these, because in many cases they allow you to avoid painful Vimscript, and to do things your way, without having to learn an ad-hoc language or to rely on plugins. This was touched on briefly in the [“Editing”](#) article of the [Unix as IDE](#) series.

## Choosing your shell

By default, vi will use the value of your SHELL environment variable as the shell in which your commands will be run. In most cases, this is probably what you want, but it might pay to check before you start:

```
:set shell?
```

If you're using Bash, and this prints /bin/bash, you're good to go, and you'll be able to use Bash-specific features or builtins such as `[[` comfortably in your command lines if you wish.

## Running commands

You can run a shell command from vi with the `! ex` command. This is inherited from the same behaviour in `ed`. A good example would be to read a manual page in the same terminal window without exiting or suspending vi:

```
:!man grep
```

Or to build your project:

```
:!make
```

You'll find that exclamation point prefix `!` shows up in the context of running external commands pretty consistently in vi.

You will probably probably need to press Enter afterwards to return to vi. This is to allow you to read any output remaining on your screen.

Of course, that's not the only way to do it; you may prefer to drop to a forked shell with `:sh`, or suspend vi with `^Z` to get back to the original shell, resuming it later with `fg`.

You can refer to the current buffer's filename in the command with `%`, but be aware that this may cause escaping problems for files with special characters in their names:

```
:!gcc % -o foo
```

If you want a literal `%`, you will need to escape it with a backslash:

```
:!grep \% .vimrc
```

The same applies for the `#` character, for the *alternate buffer*.

```
:!gcc # -o bar  
:!grep \# .vimrc
```



And for the `!` character, which expands to the previous command:

```
:!echo !
:!echo \!
```

You can try to work around special characters for these expansions by single-quoting them:

```
:!gcc '%' -o foo
:!gcc '#' -o bar
```

But that's still imperfect for files with apostrophes in their names. In Vim (but not vi) you can do this:

```
:exe "!gcc " . shellescape(expand("%")) . " -o foo"
```

The Vim help for this is at `:help !`.

### Reading the output of commands into a buffer

Also inherited from `ed` is reading the output of commands into a buffer, which is done by giving a command starting with `!` as the argument to `:r`:

```
:r !grep vim .vimrc
```

This will insert the output of the command *after* the current line position in the buffer; it works in the same way as reading in a file directly.

You can add a line number prefix to `:r` to place the output after that line number:

```
:5r !grep vim .vimrc
```

To put the output at the very start of the file, a line number of 0 works:

```
:0r !grep vim .vimrc
```

And for the very *end* of the file, you'd use `$`:

```
:$r !grep vim .vimrc
```

Note that redirections work fine, too, if you want to prevent `stderr` from being written to your buffer in the case of errors:

```
:$r !grep vim .vimrc 2>>vim_errorlog
```

### Writing buffer text into a command

To run a command with standard input coming from text in your buffer, but *without* deleting it or writing the output back into your buffer, you can provide a `!` command as an argument to `:w`. Again, this behaviour is inherited from `ed`.

By default, the whole buffer is written to the command; you might initially expect that only the current line would be written, but this makes sense if you consider the usual behaviour of `w` when writing directly to a file.

Given a file with a first column full of numbers:

```
304 Donald Trump
227 Hillary Clinton
3  Colin Powell
1  Spotted Eagle
1  Ron Paul
1  John Kasich
1  Bernie Sanders
```

We could calculate and view (but not save) the sum of the first column with `awk(1)`, to see the expected value of 538 printed to the terminal:

```
:w !awk '{sum+=$1}END{print sum}'
```

We could limit the operation to the faithful electoral votes by specifying a line range:

```
:3,$w !awk '{sum+=$1}END{print sum}'
```

You can also give a range of just `.`, if you only want to write out the current line.

In Vim, if you're using visual mode, pressing `:` while you have some text selected will automatically add the `'<,'>` range marks for you, and you can write out the rest of the command:

```
:'<,'>w !grep Bernie
```

Note that this writes every *line* of your selection to the command, not merely the characters you have selected. It's more intuitive to use visual line mode (`Shift+V`) if you take this approach.

### Filtering text

If you want to *replace* text in your buffer by filtering it through a command, you can do this by providing a range to the `!` command:

```
:1,2!tr '[:lower:]' '[:upper:]'
```

This example would capitalise the letters in the first two lines of the buffer, passing them as input to the command and replacing them with the command's output.

```
304 DONALD TRUMP
227 HILLARY CLINTON
3 Colin Powell
1 Spotted Eagle
1 Ron Paul
```



1 John Kasich  
1 Bernie Sanders

Note that the number of lines passed as input need not match the number of lines of output. The length of the buffer can change. Note also that by default any stderr is included; you may want to redirect that away.

You can specify the entire file for such a filter with %:

```
:%!tr '[:lower:]' '[:upper:]'
```

As before, the current line must be explicitly specified with . if you want to use only that as input, otherwise you'll just be running the command with no buffer interaction at all, per the first heading of this article:

```
.:!tr '[:lower:]' '[:upper:]'
```

You can also use ! as a *motion* rather than an ex command on a range of lines, by pressing ! in normal mode and then a motion (w, 3w, }, etc) to select all the lines you want to pass through the filter. Doubling it (!! ) filters the current line, in a similar way to the yy and dd shortcuts, and you can provide a numeric prefix (e.g. 3!!) to specify a number of lines from the current line.

This is an example of a general approach that will work with any POSIX-compliant version of vi. In Vim, you have the gU command available to coerce text to uppercase, but this is not available in vanilla vi; the best you have is the tilde command ~ to *toggle* the case of the character under the cursor. tr(1), however, is specified by POSIX—including the locale-aware transformation—so you are much more likely to find it works on any modern Unix system.

## ABOUT THE AUTHOR

# Interview with Tom Ryder

### Can you tell our readers about yourself and your blog?

I'm a systems administrator with some web development experience who caught the Unix bug in my late teens. I live in provincial New Zealand, and work for an internet services provider, mostly in the care and feeding of our Unix-like servers. I don't have much computer science education. My blog [Arabesque][1] focusses mostly on command-line tools for Unix-like operating systems, a topic that fascinates me. I've been writing it since 2012.

### How you first got involved with blogging?

I started the blog mostly to practise technical writing, and to formalise my own knowledge. At the time, I was linking some of the posts to relevant sections on Reddit. As I found that more people were reading the articles, especially the content on Vim, I began to put more effort into making the posts more

If you end up needing such a command during editing a lot, you could make a generic command for your private bindir, say named upp for uppercase, that forces all of its input to uppercase:

```
#!/bin/sh
cat -- "${@:--}" |
tr '[:lower:]' '[:upper:]'
```

Once saved somewhere in \$PATH and made executable, this would allow you simply to write the following to apply the filter to the entire buffer:

```
:%!upp
```

The main takeaway from this is that the scripts you use with your editor don't have to be in shell. You might prefer Awk:

```
#!/usr/bin/awk -f
{ print toupper($0) }
```

Or Perl:

```
#!/usr/bin/env perl
print uc while <>;
```

Or Python, or Ruby, or Rust, or ...

Incidentally, this “filtering” feature is where vi's heritage from ed ends as far as external commands are concerned. In POSIX ed, there isn't a way to filter buffer text through a command in one hit. It's not too hard to emulate it with a temporary file, though, using all the syntax learned above:

```
*1,2w !upp > tmp
*1,2d
*0r tmp
*!rm tmp
```

generally useful. I'd found many good blogs about computers and was mostly attempting to emulate the things I liked about them. A lot of the posts boil down to explaining and demonstrating things in a more discursive way than the manual pages do. Not that we shouldn't be reading manual pages, but a good tutorial for an initial approach to a program never hurts, if only to orient you toward one possible way of thinking about any given problem you have, or to get a better idea of the features available to you in a program.

### What's the best thing a blogger can give to his readers?

I don't really know how to answer that--the posts would be a lot more popular if I did! There are definitely some standards I try to meet in writing, but I don't know of a magic bullet. I write about the things that interest me on the Unix command line, in the hopes that others will be interested too. Each post is really just casting an idea out into the void. They're not really marketed or targeted. I'm glad if people read them and find them useful, but I'd probably still be writing the same content if I only had four or five readers a day.

### Everyone has a favorite/least favorite post. Name yours and why?

I'm fond of the "Vim Koans" page. I enjoy playing with the idea of computers as a kind of modern mysticism, and I'm fascinated with Eastern esotericism in general, though I'm more interested in Vedantic Hinduism than Buddhism. The inspiration came from ["The Rootless Root" or "The Unix Koans of Master Foo"]<sup>[2]</sup> on Eric Raymond's website.

The concrete techniques and tools of computing are generally not too hard to describe in plain language, given a sufficient technical vocabulary, but anything to do with a philosophy of computing or an approach to design or good general practice can be a lot harder to explain. A lot of the time you can really only grasp it by example and demonstration. The masters in zen stories often worked the same way, demonstrating an abstract concept in concrete terms, and frequently with a sense of humour.

### What do you think what makes Unix so beloved by programmers?

Unix has a spirit of trusting the user to know what they are doing, and inplacing a high value on freedom. This isn't necessarily freedom in the pure ethical sense that Richard Stallman might advocate, staunchly enforced by legal structures; it's more of a creative kind of freedom, providing you with the tools without trying to prescribe or even to predict exactly how they're going to be used. That approach contributes to an ethos of the system staying out of your way: setting up as few artificial roadblocks as possible to getting what you want done. The idea of open source is an extension of this, which explains why Unix-like operating systems fit so neatly into its mould--while there are certainly ethical and commercial benefits to the use and production of open source software, for a lot of technical people it really more boils down to removing artificial barriers to solving problems. To programmers who are used to dealing with many abstractions and trying to solve complex problems by a "divide and conquer" strategy, that approach has a lot of appeal, especially where it extends to \*toolsmithing\*--providing you with a means to customise and create your own tools, whether a patch or extension to your editor that you use every day, or a throwaway Awk metaprogram creating a few hundred lines of shell script to solve a one-off task.

### What is your advice to anyone who wants to advance their UNIX knowledge?

Just find something you love to do (especially if it's creating something), and do it with your favourite flavour of Unix, whether that's programming, writing, typesetting, desktop customization, or any of the hundreds of other things for which there are so many time-tested tools available to you. Don't sweat too much whether you're approaching learning "the right way"; read what others in the community are doing with the tools, try them out, and see how they fit with the way you like to work. Discard anything that doesn't. As soon as you find something you don't like that's inefficient, or wonder

if there's a better way to do something, or want something to run faster or more automatically--or even if you're just curious as to why a program runs a certain way--explore that. Trust your instincts! As you develop in whatever you're doing, you'll find yourself learning more and more about the toolset and trying different tools, and before you know it you'll be wanting to \*customise\* those tools, and then eventually to write your own. From that point, everything clicks. With documentation and source code everywhere, you realise you can learn whatever you need to about the system whenever you need to, and that all it takes is motivation and time. Understanding the system will stop seeming like the preserve of a priesthood of genius computer scientists. You won't feel the need to emulate anybody, nor to prove yourself, and you'll truly make your sense of expertise your own. It's a great feeling, and is worth striving for. This is, I think, the best way to learn anything new in depth, not just computers--find a way to link it with something you already care about and love to do. That way, it doesn't even really feel like work.

### What is your favourite OS and why?

I run Debian GNU/Linux both at home and at work, and I tinker a lot with the major BSD systems in virtual machines, particularly to test code for compatibility. I'm also fond of OpenBSD. I am not much of a zealot in terms of choosing which Unix to use, though, generally--whatever gets the job done and works reliably is probably all right with me.

### What is the future of UNIX in general or your favourite OS? What do you think?

I used to be interested in the idea of spreading Linux usage to the general population and making it accessible and "ready for the desktop", but I don't worry about that much now; I don't think it's important to have mass-market appeal for the system. I think rather than focussing on mass-market appeal which would probably just adulterate what makes Unix unique, advocacy should instead focus on reaching out to the kind of person who finds this approach to computing valuable and worthwhile, because they will always be out there. While I have no idea what the future holds for Unix as a system specifically, I'm pretty confident that the ideas that made it successful will endure permanently, as will the community that values those ideals.

### Do you have any specific goals for the rest of this year?

Fnord! You are not cleared for this information!

There's a big queue of article topics in the pipeline. We'll see how many actually get published...

[1]: <https://sanctum.geek.nz/arabesque/>

[2]: <http://catb.org/esr/writings/unix-koans/>

# Interview with Benjamin Wright

**Benjamin Wright is a practicing attorney based in Dallas, Texas, focusing on technology law. He serves as a senior instructor at the SANS Institute, teaching a 5-day course titled 'Law of Data Security and Investigations.' Through that course, Mr. Wright has taught thousands of students across the world. Moreover, he chairs SANS Institute's annual Data Breach Summit and advises diverse clients, both in the US and outside the US, on privacy, electronic commerce and data security law.**



## **How you first got involved in trainings?**

I am a lawyer in private practice. My background is business law, which means that I have much experience writing and interpreting contracts and policy documents. I have spent most of my career focused on technology law.

In the 1990s, I wrote books on the law of electronic commerce. It was a pioneering topic in those days. People were beginning to do business electronically and without paper. The replacement of paper raised questions about evidence, records and signatures.

My work in electronic commerce attracted clients and many opportunities to speak and teach.

My experience in teaching led to a relationship with the SANS institute in 2002. My knowledge of electronic commerce law matched well with the concerns of information security and digital forensics.

My relationship with the SANS Institute has been excellent in my career. By teaching at SANS, I meet some of the smartest people in the world of information security and forensics, and they teach me tips, stories and ideas. This not only makes me a better lawyer for my clients but also improves my course.

## **What's the best thing an instructor can give to his students?**

One of my greatest objectives is to help professionals in information security and forensics be cautious in their choice of words as they write reports and policies. Words matter. But often, people with a technical background don't have a lot of training on how to choose their words carefully. Increasingly, however, the words they write into emails and reports and policies have legal implications. Those words can be reviewed by legal authorities such as courts or regulators.

Additionally, one of my goals as an instructor is to help professionals evaluate the quality of digital evidence, whether it's an evidence of a data breach or forensic evidence used in some other kind of investigation. In my experience, many people leap to conclusions about digital evidence without carefully evaluating the quality of the evidence. It is easier to see a minor trace of information and interpret it in the wrong way. Some people, for example, are very eager to interpret a security incident as resulting in an actual "breach" of data security. However much incidents happen all the time, not every incident is a breach of security. The evidence from an incident must be evaluated carefully and skeptically before anyone reaches a conclusion that something like a data security "breach" has occurred.

**Could you tell more about OSINT Law. What is legal?**

I teach professionals how to read the legal terms and conditions that apply as they collect any evidence or open source intelligence. When a professional goes into a website such as Facebook, legal terms and conditions apply. Surprisingly, those terms and conditions may limit the ability of the professional to look for information or capture evidence. Similarly, when a professional opens a mobile app and uses it to capture evidence about some other user, an end user license agreement (EULA) applies. Many forensic professionals lack the training necessary to read and understand end user license agreements as they apply to their ability to gather evidence. An end user license agreement may forbid or restrict the capture of evidence.

**What is the most difficult issue in InfoSec law today?**

One of the most difficult topics in information security law today is how to judge whether an organization has done enough to protect the information it controls.

Some authorities suggest that an organization is expected to apply "reasonable controls" to protect data. However, there is much confusion about what constitutes "reasonable controls" in any given situation. And in truth, information security is so difficult that even if you have reasonable controls, you can still be hacked. Some regulators and legal authorities seem to take the position that if you've been hacked, then you are necessarily unreasonable and you must be held liable. So in other words, there's no way an organization will legally win.

My argument is that, if an organization applies "professional attention" to the protection of data, then it should not be liable if it is hacked or the data is breached.

It's like going to a doctor. If you have a disease and go to the doctor, the doctor cannot guarantee that you will get cured. But if you die of the disease, the doctor will not necessarily be held liable by law. So long as the doctor applied "professional attention" to your disease, then the doctor is deemed to have done what was necessary and should be held blameless in the law.

My suggestion is a similar standard should apply to organizations that hold sensitive data like credit card numbers. If they have professionals working on the problem of protecting data, then that should be enough. They should not then be held liable if they are hacked, or the data is breached.

**Do you have any specific goals for the rest of this year?**

One of my primary goals this year is to lead a successful two-day conference on data breaches. We at the SANS Institute call it the "Data Breach Summit," which will be held in Chicago in September.

Last year, we held the first data breach Summit, which was a one day event. It was such a success that we decided to extend it to two days. We are looking forward to pull together the most knowledgeable people across the world to participate intensively in the two-day conversation about how to respond to cyber crises and data breaches. We have requested for an input from government, law enforcement, cyber insurance companies, as well as privacy officers and Chief Information Security Officers.

More information is available at these locations:

<http://benjaminwright.us>

<https://plus.google.com/u/0/+BenjaminWright1>



# Infrastructure Management

Dear Readers,

As with many businesses, big or small, private or parastatal corporations, all have a framework (INFRASTRUCTURE) which establishes key features that defines the basic structures and fundamentals that make every organization what it is today. It's understanding why you are there and what is required of you to make the task at hand manageable. In our world, which I refer to as the outer web of information on technology which encircles us to a cyber-life, surrounds us with many components, policies, equipment, data and human resources. Reaching overall effectiveness makes you comprehend more about your purpose as a whole no matter the role you play or the task at hand. Every employee should not only understand their job criteria but also the impact they have and how efficient and valuable they are to the company. This doesn't only apply when they are executing their task at hand but also understanding what is it they have to do. Upon realizing this, they'll be able to figure out how their infrastructure comes to play. I.e. fundamentally establishing a set frame work that enables the company understand all it takes to be successful in this IT world.

Maintaining stability of a workstation environment gives you an understanding on how (MANAGEMENT) of key features sets that framework to an everyday protocol, ensuring facilitation and teamwork. If you love IT as much as I do, then what you do best will be a step towards a better business. Don't complicate what you already know. Make it simple as showing up for work and making sure your job criteria is fulfilled. However, you must understand your job requirements. Surprisingly, understanding what's at your desk may not be all that you need to know. It's worth to know how you can improve it, how your company works and what else to be done to maintain the best practices. The work environment should regularly establish work-related meetings; in case of any updates or changes which have to be made, they are applied in unity. As it may be, there is a possibility of most general areas in "IT" that may or may not be out-sourced, the primary reason for protecting sensitive data. The one thing which will always be considered as long as there is "IT" is cyber security. Cyber security is fairly new. It does not guarantee you or your business will always be secured. And importantly, it is not if you get breached but when your system will face cyber-attack. You can surf the web for the meaning of "Infrastructure Management". In doing so, you will get work environment guidelines, protocol and procedures. The main goal behind this is hopefully to get you to understand what needs to be addressed, the steps to be taken, and basically, to

get everyone on the same page. Additionally, in case you don't know, they strive to make sure you know who to go to.

However, I'm taking a different approach. I'm looking at this in a "how would I do it way" to be able to acknowledge certain things. Thus, I will start by addressing some issues. If you are in this business and depending on the size of the company, whether it's HID cards, proximity or whatever the case regardless, know what they understand by these things. I'm not saying you should stop what you are doing, learn coding and master the seven layers to be a cyber-warrior. If you are at the top of management in your business or company, you should have a very skilled network systems administrator or a cyber-security pro. He or she must be well aware of his mandate. However, you must still prove your worthiness for the company. Earlier, I mentioned the key cards, though I won't get into much detail about them. Strive to know what they are and how they work. Usually, big companies spend money on highly secured systems., but others can be breached from a few feet away. Take necessary steps to avoid having risk of being breached. When you go for lunch, be careful where you keep your card. You can all agree to a secured type of storage with a safe guard like signal blocking during an employee meeting.. Discuss the essence of short keys in locking your computer if needed to to reduce the probability of breach. Even with company email, there are better alternatives that can keep employees' conversations private. So, why does the Government and other officials use print email? Print email can be easily destroyed and it's about time you adopted this form of communication. There are different ways to shred documents. In destroying confidential documents, always ensure no information can be traced, even if it means throwing water in the shredded material.

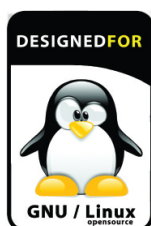
A small thing when applied correctly can help out in a big way. With software like ransom ware, you can exploit its features to block access to your firm's infrastructure. Let's hope during your meeting, there is a secured way of backing up data. Also, learning to unplug connected devices can save you down time. Mostly, it's the things we can easily control but we are naïve to see. It's worth noting that the most highly secured systems are not on-line. So, there is hope after all and I hope you can do it! Every opinion counts. Speak up; don't just listen to the one who seem to belittle your idea of securing your system.. Thank you and good luck in ensuring a better infrastructure management.

Best regards,  
Randy Ramirez  
(cyberlife25)



## Rack-mount networking server

Designed for BSD and Linux Systems



Designed. Certified. Supported

Up to **5.5Gbit/s**  
routing power!



### KEY FEATURES

- ▶ 6 NICs w/ Intel igb(4) driver w/ bypass
- ▶ Hand-picked server chipsets
- ▶ Netmap Ready (FreeBSD & pfSense)
- ▶ Up to 14 Gigabit expansion ports
- ▶ Up to 4x10GbE SFP+ expansion

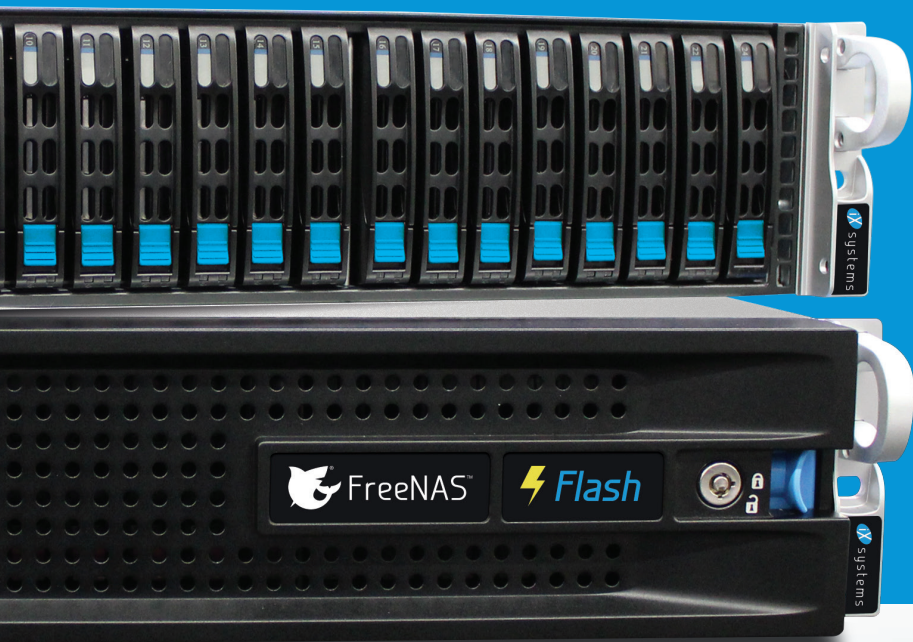


### PERFECT FOR

- ▶ BGP & OSPF routing
- ▶ Firewall & UTM Security Appliances
- ▶ Intrusion Detection & WAF
- ▶ CDN & Web Cache / Proxy
- ▶ E-mail Server & SMTP Filtering

contactus@serveru.us | www.serveru.us  
8001 NW 64th St. Miami, FL 33166 | +1 (305) 421-9956





# IS AFFORDABLE FLASH STORAGE OUT OF REACH?

**NOT ANYMORE!**

## IXSYSTEMS DELIVERS A FLASH ARRAY FOR UNDER \$10,000

**Introducing FreeNAS® Certified Flash.** A high performance all-flash array at the cost of spinning disk.

### KEY ADVANTAGES

- ⚡ 10TB of all-flash storage for less than \$10,000
- ⚡ Unifies SAN/NAS for block and file workloads
- ⚡ Runs FreeNAS, the world's #1 software-defined storage solution
- ⚡ OpenZFS ensures data integrity
- ⚡ Scales to 100TB in 2U
- ⚡ Perfectly suited for Virtualization, Databases, Analytics, HPC, and M&E
- ⚡ Performance-oriented design provides maximum throughput/IOPs and lowest latency
- ⚡ Maximizes ROI via high-density SSD technology and inline data reduction

The all-flash datacenter is now within reach. Deploy a FreeNAS Certified Flash array today from iXsystems and take advantage of all the benefits flash delivers.

For more information, visit [ixsystems.com/FreeNAS-Certified-Servers](http://ixsystems.com/FreeNAS-Certified-Servers) today.

